







Digitized by the Internet Archive  
in 2019 with funding from  
University of Alberta Libraries

<https://archive.org/details/Capjack1965>



Thesis  
1965  
#1

THE UNIVERSITY OF ALBERTA  
ITERATIVE ANALOG TECHNIQUES

by  
CLARENCE E. CAPJACK

A THESIS  
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE  
OF MASTER OF SCIENCE

EDMONTON, ALBERTA  
MARCH, 1965



UNIVERSITY OF ALBERTA  
FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled Iterative Analog Techniques submitted by Clarence E. Capjack in partial fulfilment of the requirements for the degree of Master of Science.



## ABSTRACT

The object of this thesis is to develop techniques for the iterative solution of problems on the analog computer. An essential requirement for iterative computation is storage. For this purpose an iterative control unit was built containing the necessary logic and solid-state switching. An analysis is made of the various types of memories that may be implemented using solid-state switching on the analog computer. The application of iterative techniques to the solution of a two-point boundary value problem is given for a linear second order differential equation for illustrative purposes. The generalization to more complicated, nonlinear problems is immediate since the iteration scheme remains essentially unchanged. In the serial solution of partial differential equations, the value of the function from the previous interval of time is required. A technique is developed where this storage may be accomplished with only moderate equipment requirements. This makes the serial solution of partial differential equations using an analog computer more realistic since function storage normally imposes excessive equipment requirements. Using this function storage, the serial solution to a parabolic partial differential equation is obtained.



### ACKNOWLEDGEMENTS

The author wishes to express his appreciation for the suggestions and co-operation offered by the staff members and postgraduate students. This project was done under the supervision of Y.J. Kingma, to whom the author wishes to make special acknowledgement.



# TABLE OF CONTENTS

	Page
Section 1.0 INTRODUCTION	1
1.1 Contributions of This Thesis	3
1.2 A Survey of the Chapters	5
2.0 STORAGE ON THE ANALOG COMPUTER	6
2.1 Storage On An Operational Amplifier	6
2.2 Forward and Reverse Memories	8
2.3 Electronic Comparator	10
2.4 X-Memories and O-Memories	12
2.5 Ratchet Memory	15
2.6 Accuracy of Memories	17
3.0 SOLUTION OF BOUNDARY VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS	18
3.1 Boundary Value Problem in Ordinary Differential Equations	18
3.1.1 Computer Program	20
3.2 Conclusions	22
4.0 FUNCTION STORAGE ON THE ANALOG COMPUTER	24
4.1 Programming for An Automatic Function Generator	26
4.1.1 Method 1 (Direct Simulation)	26
4.1.2 Method 2 (Indirect Simulation)	27



	Page
Section 4.1.3 Method 3 (Newton's Iterative Function Generator	
	29
4.2 Newton's Program for Nine Points	30
4.3 Function Storage Using Nine Points	36
4.3.1 The X-Memory	36
4.3.2 O-Memory Control	39
4.3.3 O-Memory	41
4.3.4 Newton's Iterative Function Generator	43
4.4 Conclusions	48
5.0 SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS ON THE ANALOG COMPUTER	51
5.1 Parallel Method	52
5.2 The Serial Method	54
5.3 Solution of a Heat Problem Using The Serial Method	55
5.3.1 Block Diagram for the Analog Computer	56
5.3.2 Computer Program for Solution of Equation (5-2)	57
5.3.3 The Iteration Scheme	57
5.3.4 Mode Control (I.C. to Operate)	61
5.3.5 Real Time Plotting Scheme	65
5.3.6 Initial Condition $T(x,0)$	70
5.4 Theoretical Results	70
BIBLIOGRAPHY	78



	Page
APPENDIX A:      ITERATIVE CONTROL UNIT	80
APPENDIX B:      DIODE BRIDGE SWITCH	82
APPENDIX C:      ELECTRONIC COMPARATOR	84
APPENDIX D:      NEXUS COMPARATOR	88
APPENDIX E:      MONOSTABLE MULTIVIBRATOR	90
APPENDIX F:      SWITCHING ARRANGEMENT FOR INTEGRATORS	92
APPENDIX G:      NEWTON'S FORWARD DIFFERENCE	
INTERPOLATION FORMULA	94



# LIST OF TABLES

		Page
Table	4-1	Coefficients for Newton's Forward Difference Formula
		26
	4-2	X-Memory for Nine Point Function Storage
		39
	4-3	O-Memory for Nine Point Function Storage
		43
	5-1	Computer and Theoretical Results
		74
	5-2	Theoretical Results for $\Delta t = .01$
		75
	C-1	Switching Levels of Comparators
		86
	E-1	Monostable Multivibrators in the Iterative Control Unit
		91
	G-1	Determination of Coefficients a to c
		97
	G-2	Coefficients a to i
		97
	G-3	Values of $f^i(0)$
		98



## LIST OF FIGURES

		Page	
Figure	2-1	Schematic For Storage Amplifier	7
	2-2	Switching for Storage Amplifier	8
	2-3	Forward and Reverse Memory Symbols	9
	2-4	Operation of a Forward Memory	9
	2-5	Operation of a Reverse Memory	10
	2-6	Electronic Comparator	10
	2-7	Electronic Comparator Symbol	11
	2-8	Electronic Comparator Symbol	11
	2-9	X-Memory	13
	2-10	Operation of X-Memory	14
	2-11	O-Memory	14
	2-12	Operation of an O-Memory	15
	2-13	Ratchet Memory	16
	2-14	Cascaded Ratchet Memory	17
	3-1	Computer Program	21
	4-1	Block Diagram for Function Storage	24
	4-2	X-Memory in Sampler	25
	4-3	Automatic Function Generator	
		(direct simulation)	27
	4-4	Automatic Function Generator	
		(indirect simulation)	28
	4-5	Block Diagram for Newton's Iterative	
		Function Generator	29



	Page
Figure 4-5a Newton's Iterative Function Generator for Digital to Analog Converter	30
4-6 Function Generator Block Diagram (9 points)	31
4-7 Mode Control for Function Generator	32
4-8 Switches A and B	34
4-9 Diode Bridge Switch in an Integrator	35
4-10 Equivalent Circuit For Diode Bridge Switch	35
4-11 Block Diagram for Function Storage	36
4-12 Input to X-Memory	37
4-13 X-Memory Storing $f(L)$	37
4-14 X-Memory Symbols	38
4-15 O-Memory Control	40
4-16 Typical O-Memory	42
4-17 Compensation for Drift in O-Memory	42
4-18a Switching for the Function Generator	45
4-18b Algebraic Calculation for I.C.	45
4-18c Algebraic Calculation for Coefficients a to c	46
4-18d Computer Program for Iterative Function Generator	47
4-19 Performance of Newton's Iterative Function Generator	50



		Page
Figure	5-1	Parallel Solution of a Partial
		Differential Equation
		52
	5-2	Computer Set-Up for Station $x_1$
		54
	5-3	Integration Procedure in the Serial
		Method
		55
	5-4	Block Diagram for Solution of Heat
		Problem
		56
	5-5	Computer Diagram for Equation (5-2)
		57
	5-6	Computer Iteration Scheme
		58
	5-7	Computer Program for Mode Control
		62
	5-8	Input voltage to Comparator $N_1$
		63
	5-9	Real Time Plotting Scheme
		66
	5-10	Input Voltage to Comparator $N_3$
		68
	5-11	Generation of Initial Condition $T(x,0)$
		70
	5-12a	Iterative Solution of Boundary
		Value in O.D.E.
		77
	5-12b	Performance of Iterating Scheme
		(Input to Comparator 12)
		77
	5-12c	Solution of Heat Problem ( $\Delta t = .0625$ )
		77
	A-1	Iterative Control Unit
		80
	A-2	Contact Arrangement
		81
	B-1	Diode Bridge Switch Schematic
		82
	B-2	Measuring of Forward Resistance of
		Diode
		83



		Page
Figure	B-3 Modified Diode Bridge Switch	83
	C-1 Comparator Block Diagram	84
	C-2 Performance Test for Comparator	85
	C-3 Wiring Diagram for Comparator	87
	D-1 Nexus Operational Amplifier Used as an Inverter	88
	D-2 Wiring Diagram for Variable Dead- Zone Comparator	89
	E-1 Monostable Multivibrator for Negative Inhibit Pulse	90
	E-2 Monostable Multivibrator for Positive Inhibit Pulse	90
	F-1 Switching Arrangement for Integrators	92
	G-1 Newton's Method for Ten Points	94



## TABLE OF SYMBOLS

FM	-	Forward Memory
RM	-	Reverse Memory
FS	-	Function Switch
MM	-	Monostable Multivibrator
DPDT	-	Double Pole Double Throw
DFG	-	Diode Function Generator
DB	-	Diode Bridge
I	-	Integrator
SJ	-	Summing Junction
G	-	Grid



## 1.0 INTRODUCTION

Iterative computation represents one of the most effective implementations of an analog computer. In normal analog computing, many trial solutions are made before the final set of parameters are chosen. This trial technique may be accelerated if the solutions to the programmed equations are obtained in repetitive operation. Since the solutions are displayed visually, the correct selection of variables is more direct. The repetitive feature however itself only provides a means for the visual display of results. In iterative computation, the solutions are also obtained in repetitive operation. However the results obtained in each operate cycle are used to change some condition such as an initial condition or a parameter so as to make the next solution closer to that desired. That is, the procedure of adjusting various parameters for the solution to meet specified conditions has been automated.

The reason iterative computation may be referred to as the most effective implementation of an analog computer may be exemplified by the consideration of a two-point boundary value problem. The normal procedure would be to perform many runs, changing the required initial condition until such time that the boundary conditions were satisfied. This procedure is both lengthy and boring. This same problem using iterative techniques can be solved automatically in about 60 milliseconds.



Iterative techniques may be applied effectively to many problems on the analog computer. Some of the applications are listed below.

- (a) Boundary Value Problems. In this application an iteration is performed to find a remaining initial condition. Convergence to the required result is both fast and reliable.
- (b) Optimization Problems. In this case, some performance criterion such as the integral of the error squared of a system may be minimized by varying one or more parameters. The iterative technique also facilitates the application of Pontryagin's Maximum Principle to optimization problems on the analog computer.
- (c) Solution of Partial Differential Equations. The iterative technique may be applied to obtain the serial solution of a partial differential equation. Equipment requirements become more severe in this application because of the necessity for function storage.
- (d) Parameter Sweep. In this application iterative techniques may be applied to obtain for instance, a plot of the peak overshoot of a system as a function of  $\xi$ . The proper selection of  $\xi$  is then immediate. This same technique may also be used to solve optimization problems where a performance functional is to be minimized with respect to one parameter, say  $\alpha$ . This scheme can then be used to obtain a plot of



this functional as a function of  $\alpha$  on the x-y plotter. The optimum choice of  $\alpha$  is then immediate.

(e) Curve Fitting. Iterative techniques may be used to extrapolate a curve through a given set of points so so as to satisfy some error criterion. This is similar to the case where a performance functional is being minimized in a optimization problem.

(f) Solution of Eigenvalues. Iterative techniques may be used on the analog computer to find the first set of eigenvalues in boundary value problems in mathematical physics. The approach is similar to that for the two-point boundary value problem, only in this case the eigenvalue would be the parameter rather than the initial condition.

## 1.1 CONTRIBUTIONS OF THIS THESIS

In this thesis the application of iterative techniques to an analog computer are investigated using the Pace 231-R general purpose computer. An iterative control unit was built containing the necessary logic and solid-state switching required for iterative computation. The implementation of solid-state switching enabled iterative computations to be performed at higher speeds with improved accuracy as compared to normal relay switching.

The first section deals with the development of solid-state switching techniques to perform storage using the



analog computer. An analysis of the various types of memories and their respective performance is given.

The iterative solution of a two-point boundary-value problem was applied to a second order ordinary differential equation for illustrative purposes. The generalization to more complicated problems, for instance, nonlinear problems is immediate since the iteration scheme remains essentially unchanged. A very significant application of the iterative solution of a two-point boundary value problem is in the application of Pontryagin's Maximum Principle to optimization problems on the analog computer.

The major objective of this thesis was to perform the serial solution of a partial differential equation, using solid-state switching. The serial solution of a partial differential equation however imposes severe equipment requirements because of the need for function storage. A method was developed where the equipment requirement was reduced from seventy amplifiers to just twenty-five amplifiers for an automatic function generator using data stored for nine equally spaced intervals of the independent variable. With the development of mode control using field effect transistors, this requirement may be reduced to just fifteen amplifiers for parabolic curve fitting using any number of stored points. The program of extrapolating the function through the stored points is referred to as Newton's Iterative Function Generator. Since this function generator



requires but a moderate number of amplifiers for function storage, this scheme can be applied to smaller computer installations, for instance the race TR-48 computer. An illustration of the application of the serial method to the solution of a parabolic partial differential equation is given complete with an analysis of the errors involved.

## 1.2 A SURVEY OF THE CHAPTERS

Chapter 2      The technique for implementing storage on the analog computer using solid-state switching is developed. A description of the various types of memories that are used in iterative analog computation is given.

Chapter 3      The iterative solution of a two-point boundary value problem is given. A second order ordinary differential equation is used for illustrative purposes.

Chapter 4      A technique is developed for the storage of a function with only moderate equipment requirements. A program is given for an automatic function generator using nine stored points.

Chapter 5      Using the function storage developed in Chapter 4, the serial solution to a parabolic partial differential equation is given. An analysis of the errors encountered in this method is also given.



## 2.0 STORAGE ON THE ANALOG COMPUTER

The basic requirement in the application of iterative techniques to problems on the analog computer is that of storage. There are a variety of ways in which this may be accomplished. One is the use of a Memory Wheel as described by Kozak (reference 10). However this technique is mechanically cumbersome and synchronization with a computer in repetitive operation becomes a serious problem.

A second technique that can be employed is that of Hybrid Computation. A small digital computer could perform necessary storage as well as some sub-routines. Such a facility however was not available.

The third technique and the one used in this investigation is storing a charge on a capacitor across a switched amplifier. This technique lends it self very well to cases where the storage requirements are moderate. Another favorable aspect of this scheme is that only a modest amount of extra equipment is required.

### 2.1 STORAGE ON AN OPERATIONAL AMPLIFIER

The basic mechanism employed in storage using an operational amplifier is given in Figure 2-1. When switch A is closed,

$$\frac{e_o}{e_{in}} = \frac{1}{1 + \tau D} \dots\dots\dots(2-1)$$

where  $\tau = RC$



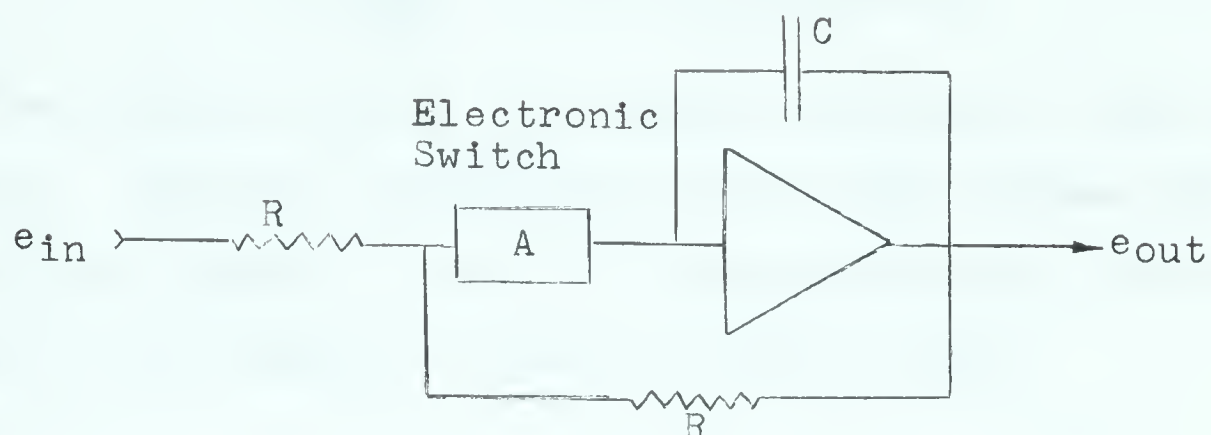


FIGURE 2-1. SCHEMATIC FOR STORAGE AMPLIFIER

If  $\tau$  is made small with respect to the time constant in the system being solved, we may approximate the expression (2-1) by,

$$\frac{e_o}{e_{in}} \approx 1$$

When switch A is open, the output voltage will remain constant. That is the amplifier will now hold or store the value at the switching instant.

Switch A is a diode bridge driven by a set of complementary flip-flops, which are triggered by a Schmitt Trigger comparator. A schematic for the switching arrangement is given in Figure 2-2.

In the amplifiers used for storage,

$$R = 100K \text{ ohms}$$

$$C = .001 \text{ ufd.}$$

Therefore  $\tau = 100 \times 10^3 \times 1 \times 10^{-9} = 1 \times 10^{-4}$  seconds.

The time required for an amplifier to track a voltage to within 1 percent error is;



$$t = \frac{4.61}{10^4} = 0.46 \text{ milliseconds}$$

This time constant can be made smaller by using a .0001 ufd. capacitor. However a compromise must be made between the ability for a memory to track a voltage accurately and the ability for this memory to hold or store a value.

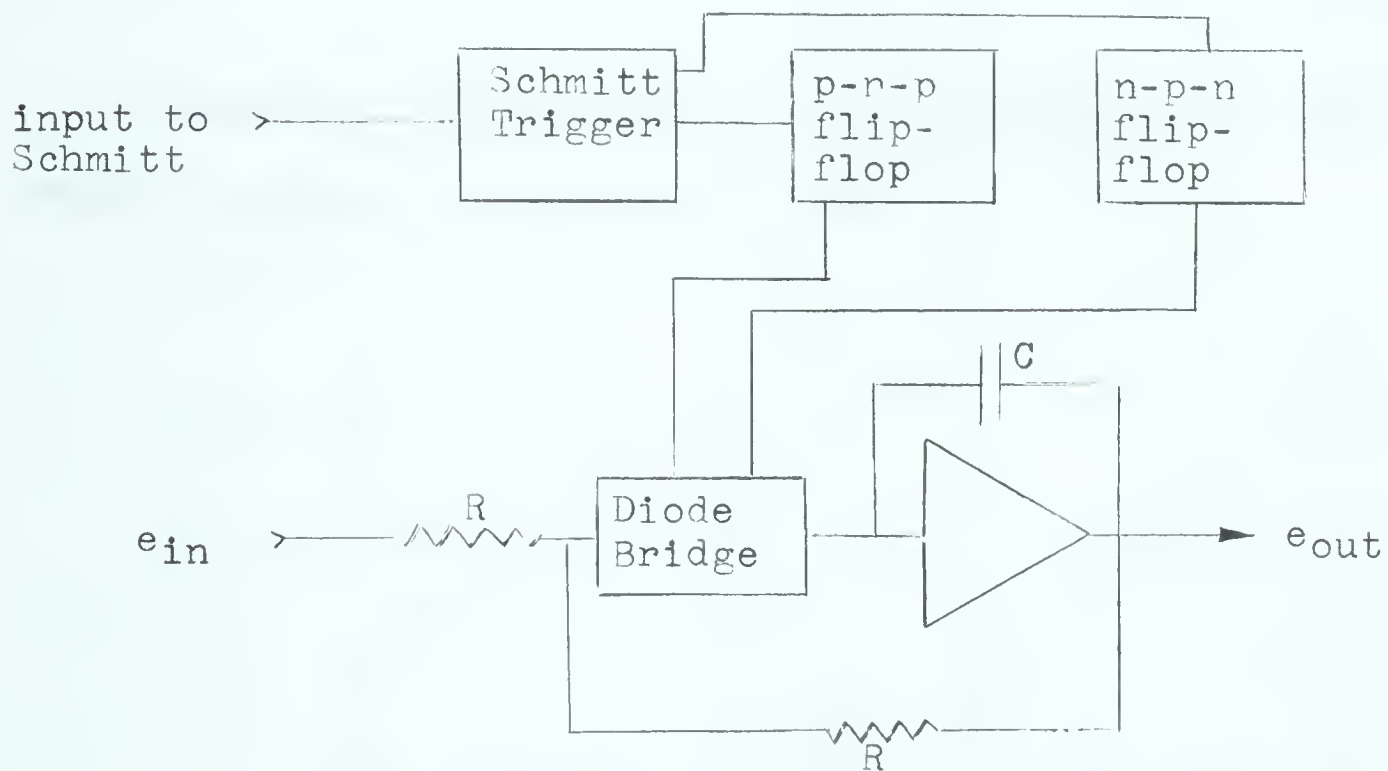


FIGURE 2-2. SWITCHING FOR STORAGE AMPLIFIER

## 2.2 FORWARD AND REVERSE MEMORIES

A forward memory (FM) is one that tracks when the computer is in the operate mode and stores or holds its value when the computer is in the hold or reset mode.

A reverse memory (RM) is the complement of a forward memory. It will track when the computer is in the reset mode and will store when the computer is in the operate mode. The symbols for each memory are given in Figure 2-3.



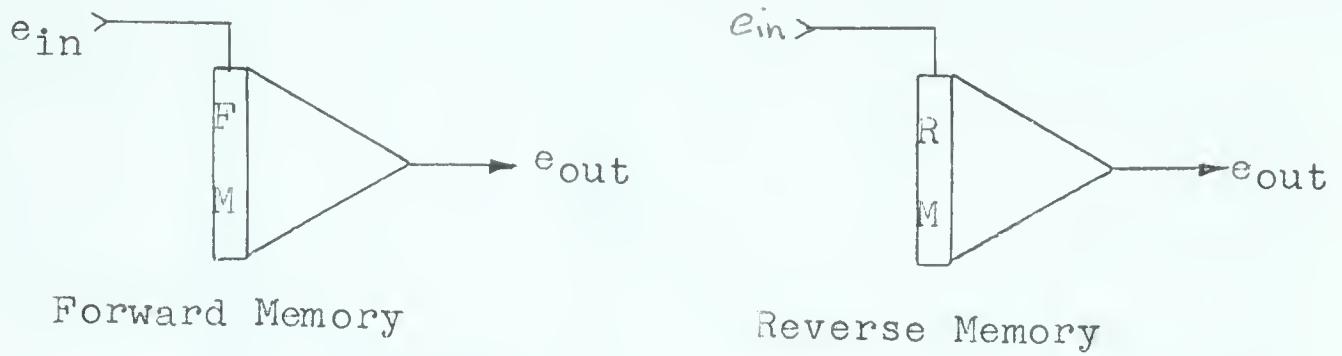


FIGURE 2-3. FORWARD AND REVERSE MEMORY SYMBOLS

A graphical explanation of the performance of each memory is given in Figures 2-4 and 2-5.

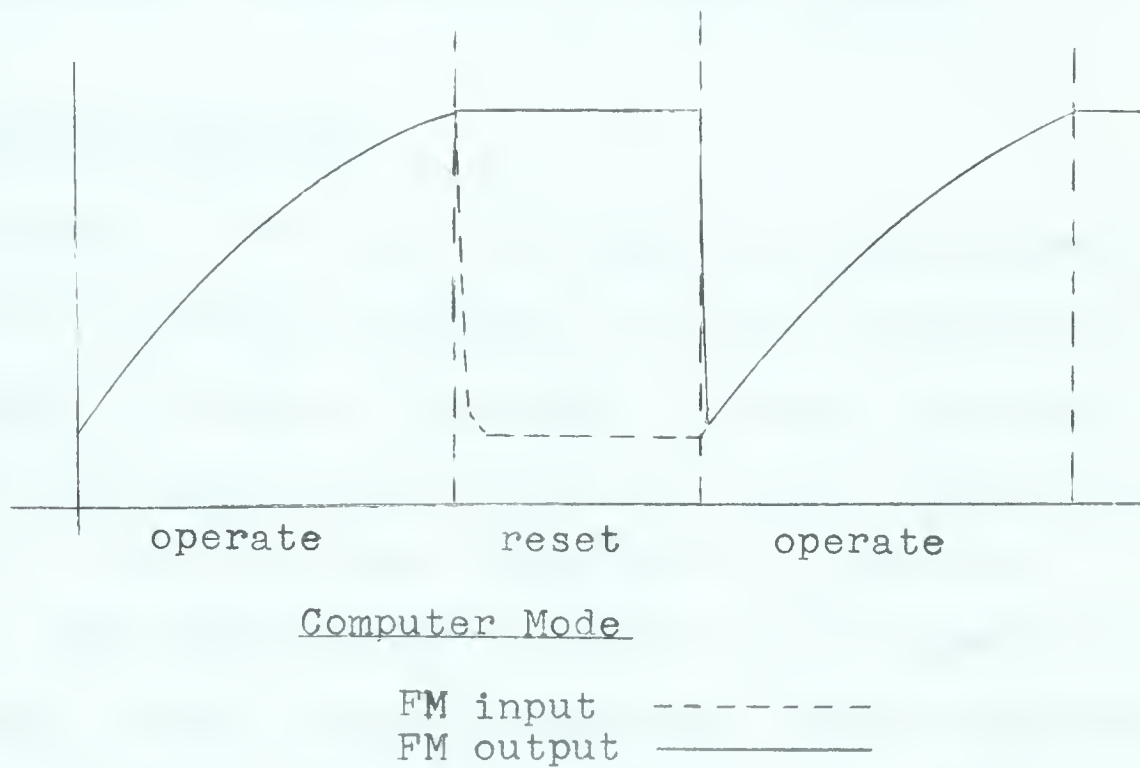


FIGURE 2-4. OPERATION OF A FORWARD MEMORY



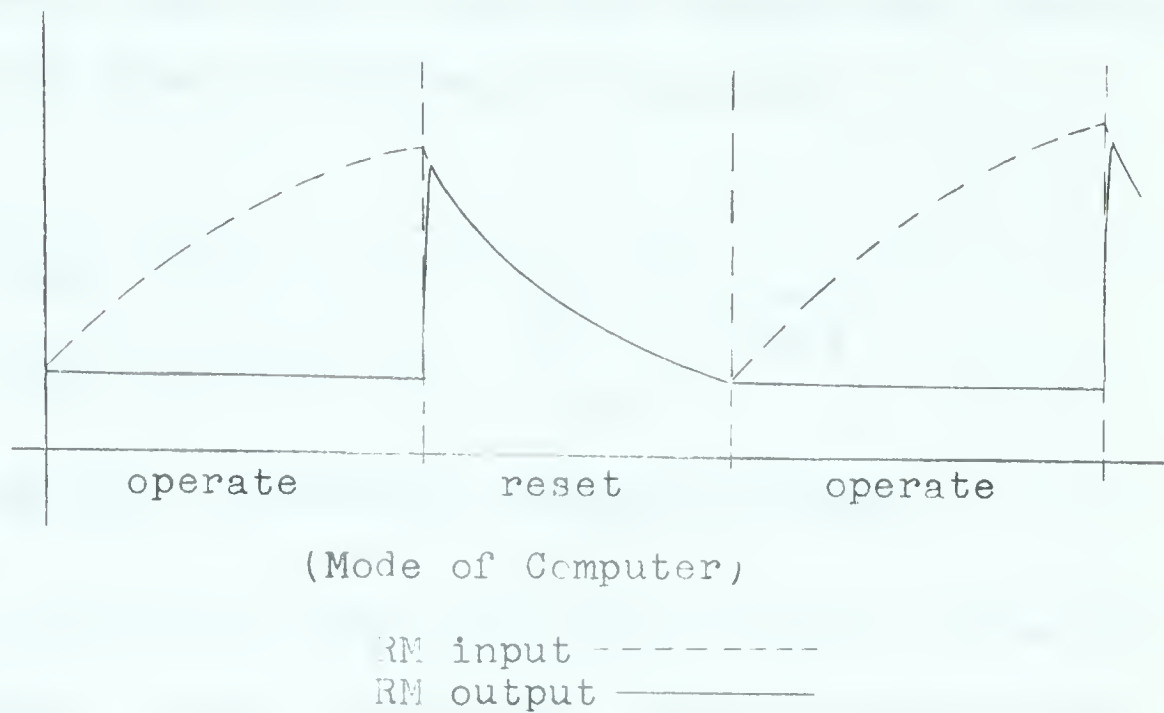


FIGURE 2-5. OPERATION OF A REVERSE MEMORY

### 2.3 ELECTRONIC COMPARATOR

The purpose of the electronic comparator in storage techniques on the analog computer is to give flexibility to the program by making it possible to switch the mode of a memory at a particular instant during the operate cycle. A schematic of the electronic comparator is given in Figure 2-6. The symbols N and S refer to the outputs of the comparator in the normal and switched state respectively.

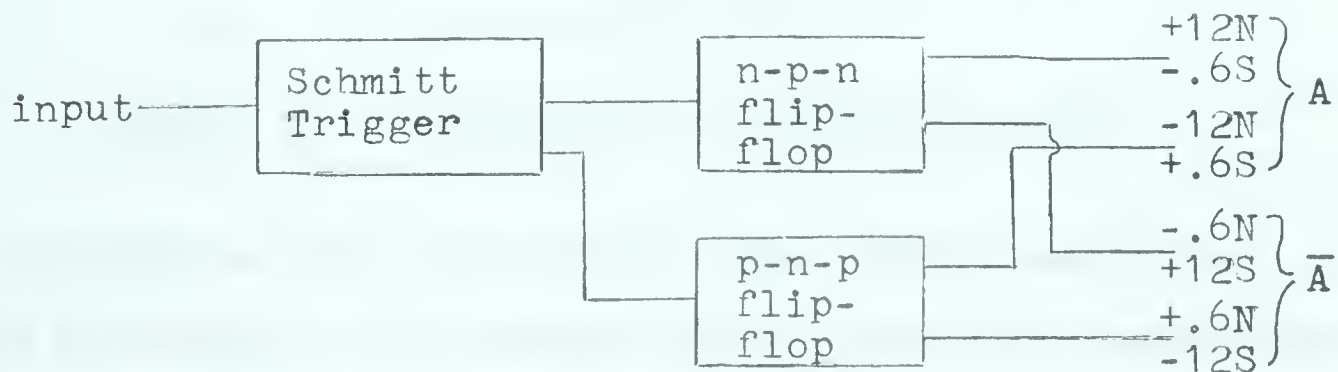


FIGURE 2-6. ELECTRONIC COMPARATOR



The symbols that will be used for designating electronic comparators are given in Figures 2-7 and 2-8.

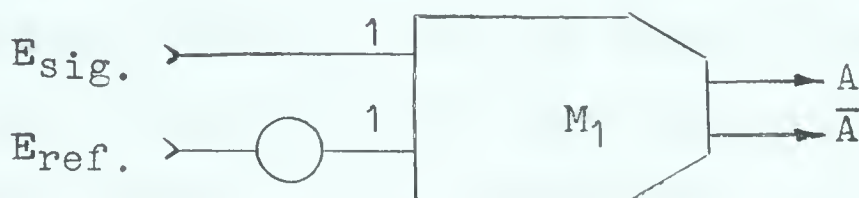


FIGURE 2-7. ELECTRONIC COMPARATOR SYMBOL

Figure 2-7 is the symbol for the electronic comparator described above. At the instant the input voltage exceeds the negative reference, the state of the comparator will change and the new output voltages will be as indicated for the switched state in Figure 2-6.

When the number of input voltages to be compared is greater than two, a unity gain amplifier is used as a summer for the comparator. The symbol that will be used in this case is shown in Figure 2-8.

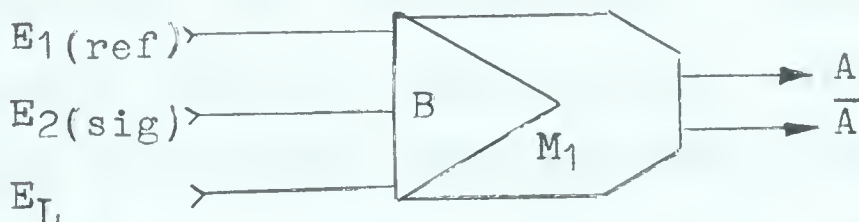


FIGURE 2-8. ELECTRONIC COMPARATOR SYMBOL

In Figure 2-8, B designates the summing amplifier while  $M_1$  refers to the number of the electronic comparator. A latch voltage ( $E_L$ ) is sometimes used which is larger



than the signal and reference voltages. The state of the comparator will then be determined only by the sign of the latch voltage. When the signal voltage is less than the reference voltage (and  $E_L = 0$ ), the output voltage from the comparator will be identical to that shown in Figure 2-6 for the normal state. At the instant the signal voltage exceeds the positive reference voltage, the output voltages A and  $\bar{A}$  will assume the values given in Figure 2-6 for the switched state.

#### 2.4 X-MEMORIES AND O-MEMORIES

X-Memories and O-Memories perform essentially the same function as the forward and reverse memories, however they are comparator controlled. An extra degree of flexibility is added when using this type of memory for storage since at any particular instant in the operate cycle a value may be stored.

The function of an x-memory is related to a forward memory in that, for the first portion of the operate cycle this memory will track its input voltage. At some particular instant during the operate cycle the mode of this memory will be changed by an electronic comparator, and this memory will now hold its last value. The symbol for an x-memory is given in Figure 2-9.



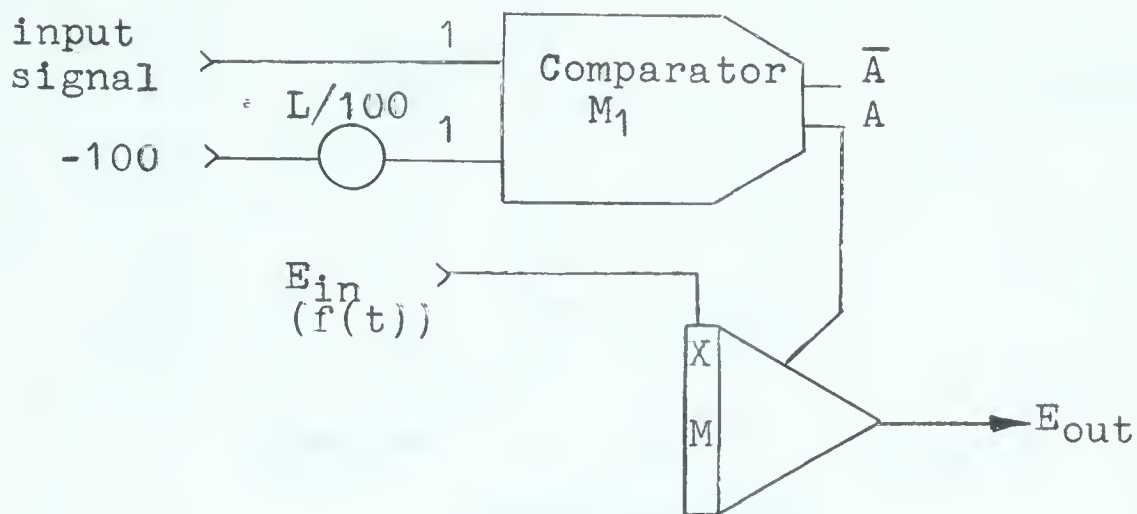


FIGURE 2-9. X-MEMORY

The x-memory in Figure 2-9 will track the input voltage until  $x = L$ , at which time its mode will change. The memory will now store or hold the value  $f(L)$  for the remainder of the operate cycle. Mode control is furnished by comparator  $M_1$  in the following fashion. When the input signal is less than the negative reference, the comparator is in its normal state. In this normal state, output  $A$  will cause the diode bridge used in the x-memory to conduct. This will cause the memory to track its input voltage. At the instant  $x = L$ , the state of the comparator will change and output  $A$  will produce voltages so as to make the diode bridge in the memory non-conducting. This in turn will cause the the x-memory to store the value of the input function at  $x = L$ . Figure 2-10 displays the operation of an x-memory.

An O-Memory is the complement of an x-memory in the same sense the reverse memory was the complement of a forward memory. The symbol for an o-memory is given in Figure 2-11.



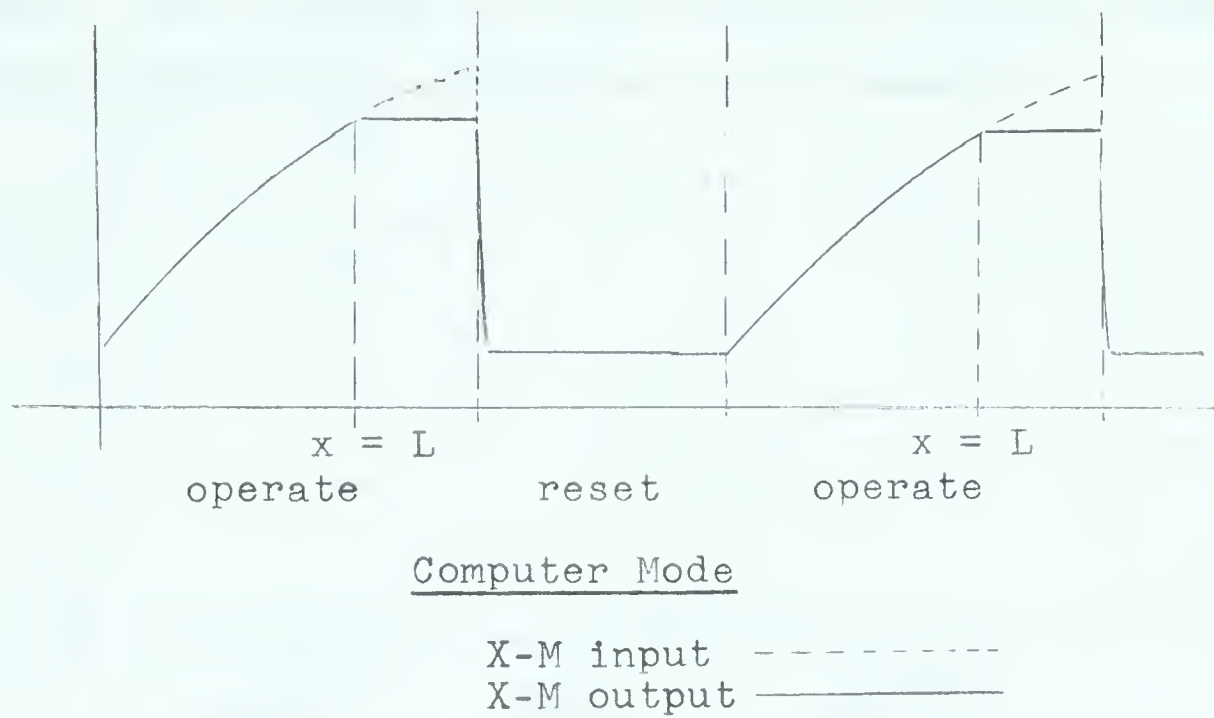


FIGURE 2-10. OPERATION OF X-MEMORY

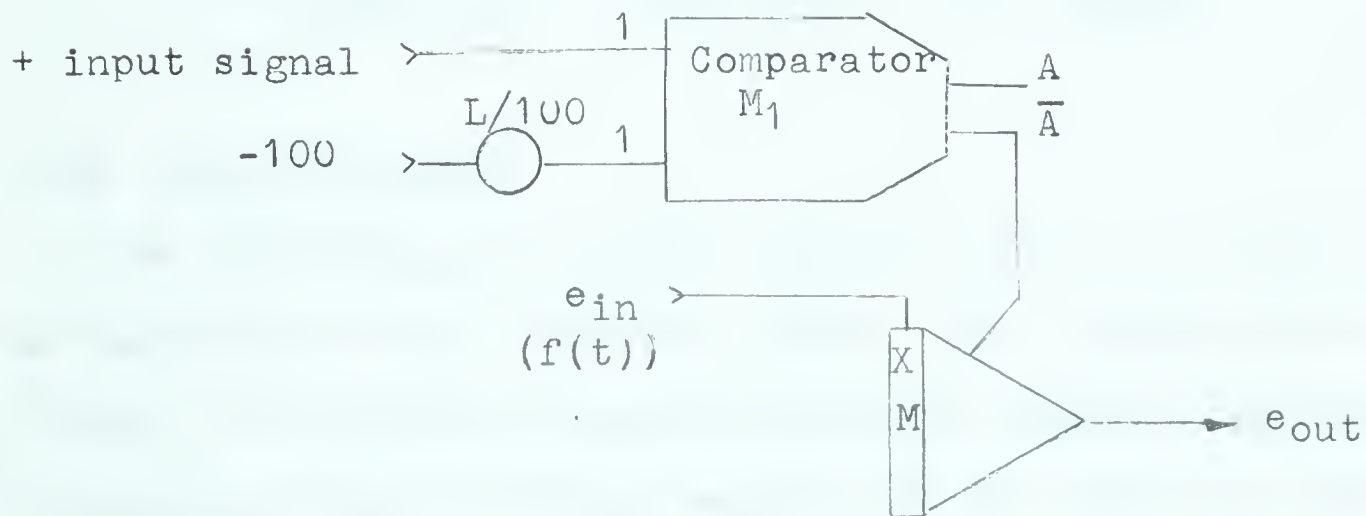


FIGURE 2-11. O-MEMORY

When comparator  $M_1$  is in the normal state, voltage  $\bar{A}$  will cause the diode bridge in the o-memory to non-conduct. In this mode the o-memory will hold or store. At the instant the input signal is equal to  $L$ , the comparator will switch its state and voltage  $\bar{A}$  will in turn cause the bridge



to conduct. The o-memory will now track its input voltage. Figure 2-12 displays the operation of an o-memory.

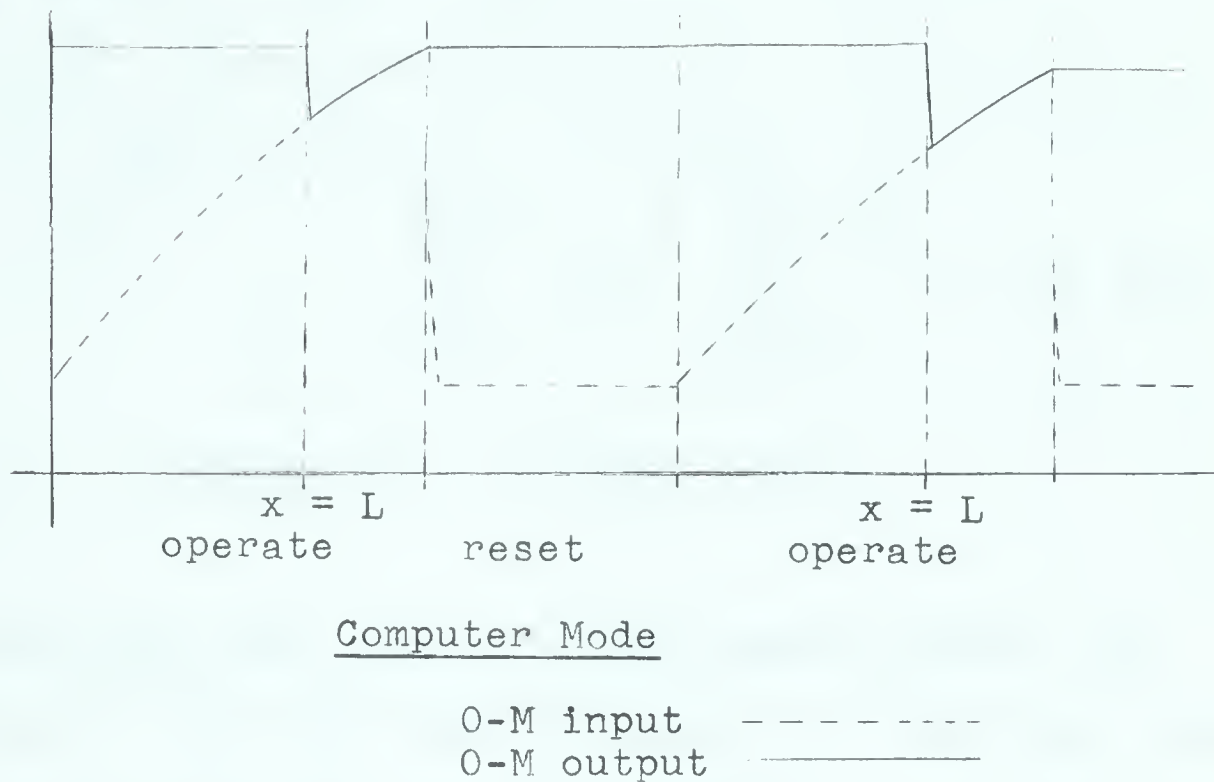


FIGURE 2-12. OPERATION OF AN O-MEMORY

## 2.5 RATCHET MEMORY

By cascading an x-memory and an o-memory a value can be remembered from a previous cycle. This technique is the key to iterative computation on the analog computer. An example where a ratchet memory can be used is in the solution of an optimization problem where some parameter or parameters are being varied in order to minimize some performance functional. The value of the performance functional from the previous iteration is required to make corrections in these parameters in order to eventually arrive at the optimum choice. A schematic of a ratchet memory is given in Figure 2-13.



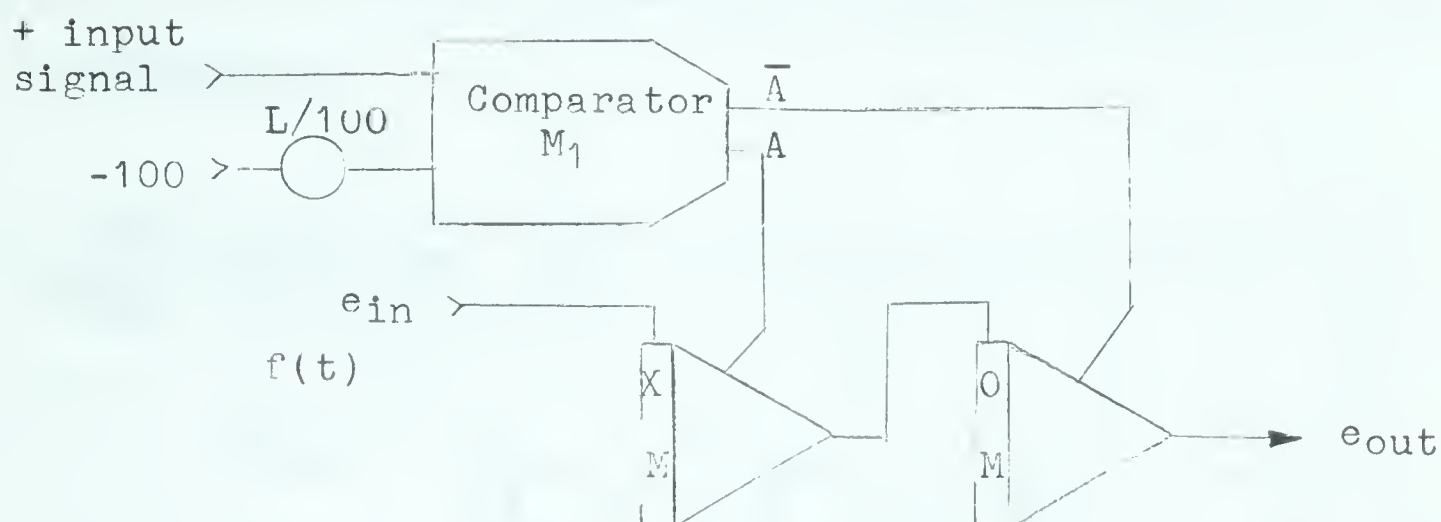


FIGURE 2-13. RATCHET MEMORY

During the first portion of the operate cycle, the x-memory will track the input voltage  $f(t)$  while the o-memory remains in the hold mode. At the instant  $x = L$ , the x-memory will store its last value, that is  $-f(L)$ , while the o-memory will now read in, or track the voltage held by the x-memory. At the completion of the operate cycle, the x-memory will again track its input voltage while the o-memory will hold or store the value  $f(L)$ . At the beginning of the next operate cycle the o-memory will still be storing the value  $f(L)$ . In this fashion a value has been remembered from a previous computation. A number of ratchet memories may be cascaded so that in the  $n$ 'th computation, values of some functional or parameter from the  $n-1$ ,  $n-2$ , ... computation may be available. The schematic for a cascaded ratchet memory is given in Figure 2-14. The figure is drawn for the  $n$ 'th computation with values of  $f(L)$  for the last two computations being made available. The generalization to the storage of



this value from more computations, for instance  $n-3$ ,  $n-4$ , ..., is immediate.

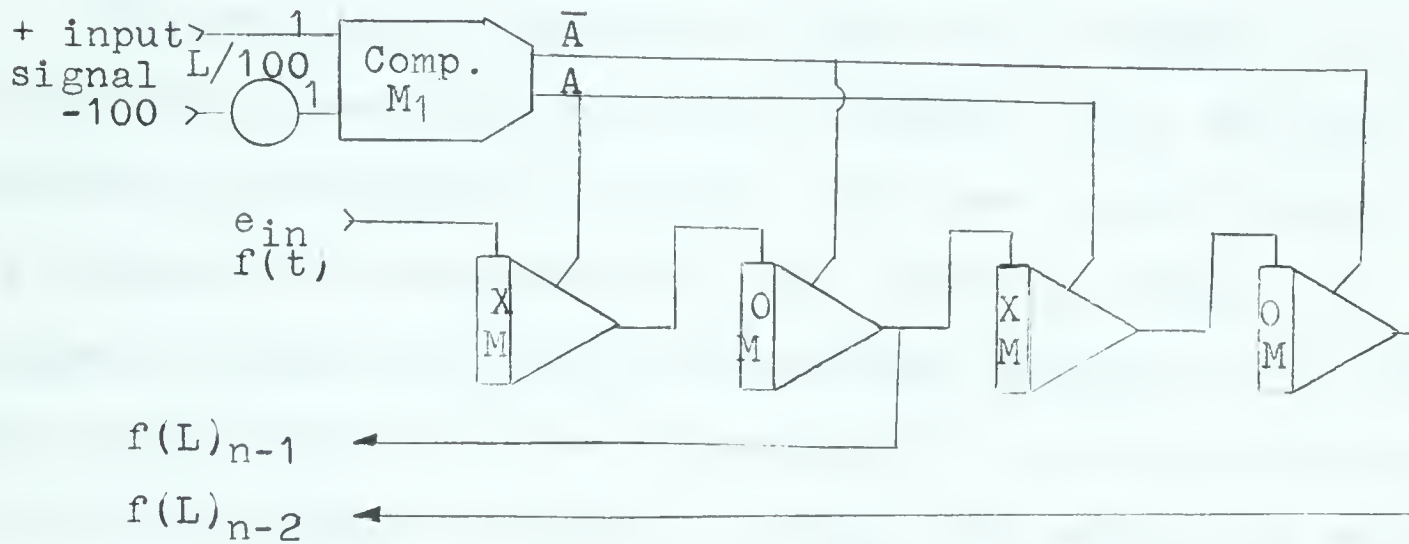


FIGURE 2-14. CASCADED RATCHET MEMORY

## 2.6 ACCURACY OF MEMORIES

A memory using a .001 ufd. capacitor for storage will track a fixed voltage to within .04 volts. A further improvement could be made by balancing the diode bridges to closer tolerances. In the hold state, drift was normally less than 0.1 volts in ten seconds. This could be further improved by placing an external bias to compensate for any bridge unbalance. In this manner drift in the hold state is less than 0.01 volts in ten seconds. When using a larger capacitor for storage, error in the track and hold modes was imperceptible.



### 3.0 SOLUTION OF BOUNDARY VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS

The technique of storage on the analog computer facilitates the automatic solution of a boundary value problem in ordinary differential equations. Only one ratchet memory is required when one boundary value is being sought. A typical problem is a trajectory problem (reference 6), where the firing angle is to be determined for a missile which is to strike a target a distance  $A$  away. One method for solving this problem is to program the applicable equations on the analog computer, then run the solution in repetitive operation. By manual adjustment, the required firing angle may be found. However, by the addition of a simple ratchet memory this process may be automated.

In the solution for an unknown boundary value in linear ordinary differential equations, the advantage of iterative computation is not evident with the exception of illustrative purposes. If the problem is made nonlinear, for instance, if air friction is taken into consideration in the missile problem, then the advantage of iterative computation becomes more evident.

In the serial solution of a partial differential equation on an analog computer, for each instant of time a two-point boundary value problem in ordinary differential equations must be solved (referred to as a nested calculation). The iteration scheme required in the solution of a partial



differential equations is practically identical to that in the solution of a boundary value problem in ordinary differential equations. An illustration of the application of iterative techniques to a boundary value problem in a second order differential equation will be given in Section 3-1.

### 3.1 BOUNDARY VALUE PROBLEM IN ORDINARY DIFFERENTIAL EQUATIONS

The technique of applying storage techniques to obtain solutions to boundary value problems will be illustrated by considering a simple second order differential equation (reference 11). The problem that will be considered is given below.

Given the O.D.E.

$$\frac{d^2y}{dx^2} + y = 0 \dots\dots\dots(3-1)$$

Find  $\frac{dy}{dx} \Big|_{x=0}$  such that the following boundary conditions are satisfied.

$$y(0) = 0$$

$$y(L) = A \quad 0 < L < \pi \quad A < 1$$

The theoretical solution may readily be obtained. This is equal to,

$$y(x) = A \sin x$$

$$y(L) = A \sin L$$

therefore  $A = \frac{y(L)}{\sin L} \dots\dots\dots(3-2)$



### 3.1.1 COMPUTER PROGRAM

The necessary scaling will first be performed. The normalized variables that will be used are as follows:

$$\frac{dy}{dx} = \frac{1}{100} \left[ \frac{dy}{dx} \right]$$

$$y = \frac{1}{100} [y]$$

$$x = \frac{1}{20} \left[ \frac{x}{5} \right]$$

The computer program is given in Figure 3-1. Scaling in the x variable is for 10 millisecond repetitive operation. Integrators 10 and 11, and amplifier 12 comprise the program for solving equation (3-1). This equation is solved in repetitive operation on the computer. For the first portion of the operate cycle, that is when  $x < L$ , amplifier 17 will track the voltage  $A - y(x)$ . At the instant  $x = L$ , comparator 1 will switch state, changing the mode of x-memory 17 and o-memory 18 so the x-memory 17 will hold the value  $A - y(L)$ . O-Memory 18 will now track this voltage. At the end of the operate cycle the mode of the memories will again change and o-memory 18 will now store the value  $A - y(L)$ . But  $A - y(L)$  serves as a practical value for the error in a given iteration. Integrator 13 will now integrate at a rate proportional to this error during the reset cycle. Amplifier 14 will now provide a better approximation for the required initial condition. This iteration process will continue until the required initial condition has been obtained.



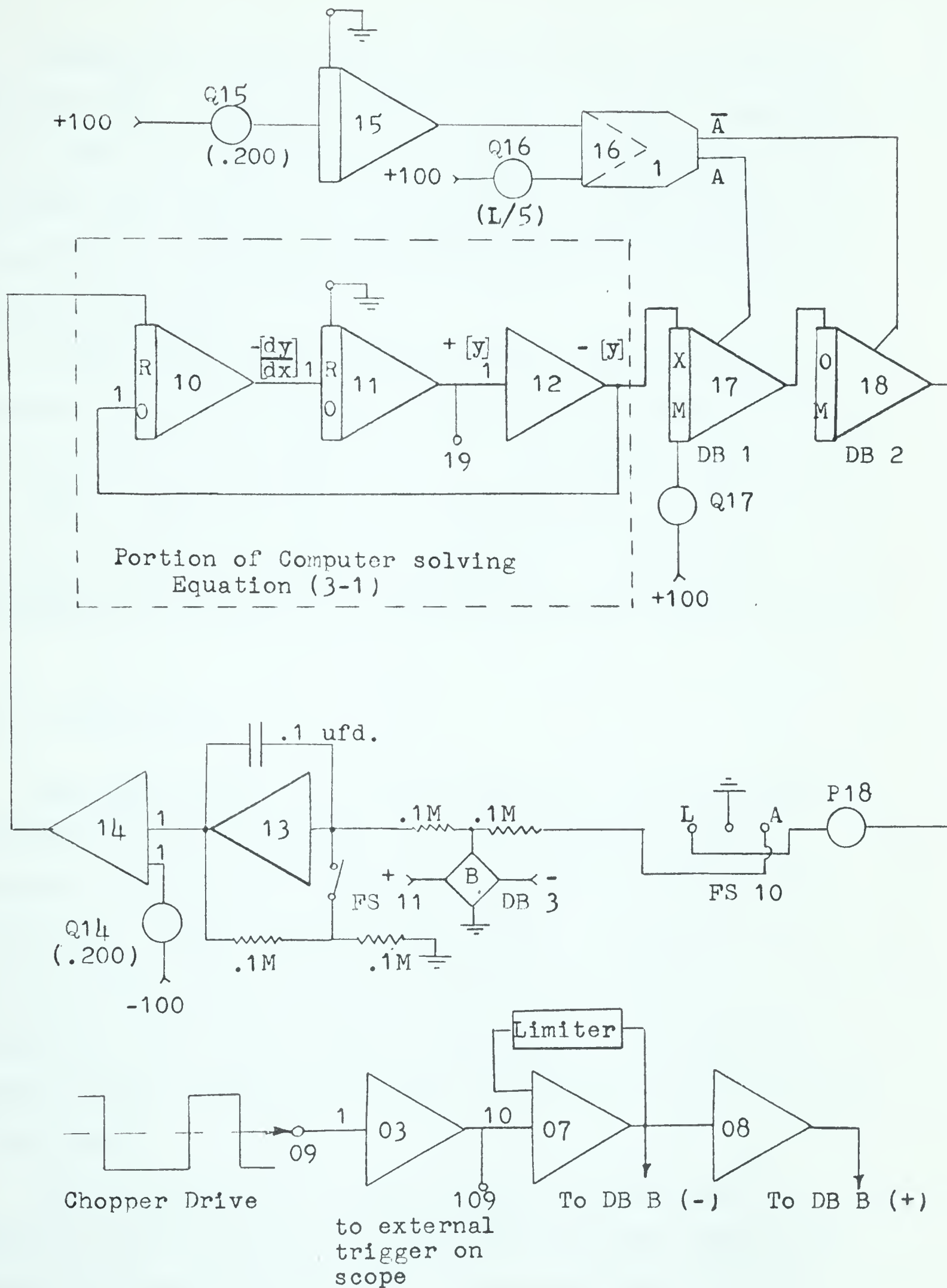


FIGURE 3-1. COMPUTER PROGRAM



During the operate cycle, integrator 13 will remain in the hold mode. This mode control is performed by amplifiers 03, 07, 08, and diode switch B. A detailed description for external mode control for integrators is given in Appendix F. Potentiometer 14 provides an initial guess for the required initial condition.

A photograph showing the successive iterations to the solution is given in Figure 5-12a. In this particular case  $L = \pi/2$  and  $y(L)$  was equal to 0.5. The results obtained gave  $A = .502$ . This is an error of less than one half of one percent, making the iterative technique of computation valuable in the solution of two-point boundary value problems.

### 3.2 CONCLUSIONS

The application of iterative techniques on the analog computer proves to be a very effective method of solving two-point boundary value problems in ordinary differential equations. If the equations are further complicated by nonlinearities, the only required addition may be a multiplier and a diode function generator. The iteration scheme however remains essentially identical to that for the linear case.

This technique of solving two-point boundary value problems may be applied to optimization problems employing Pontryagin's Maximum Principle. The possibility of using field effect transistors with Nexus Operation amplifiers for repetitive computations at a frequency of 30 kilocycles



per second has been investigated in the Electrical Engineering Department of the University of Alberta with very promising results (reference 14). This would imply that iterative computations at a frequency of at least 10 kilocycles could easily be performed. This would present an opportunity for the effective application of Pontryagin's Maximum Principle to "in-line" optimization problems on the analog computer.



#### 4.0 FUNCTION STORAGE ON THE ANALOG COMPUTER

Function storage is required when the serial solution of a partial differential equation is obtained using an analog computer. In this particular case, the value of the solution from the previous interval of time is required for obtaining the solution for the following interval of time. The need for curve storage occurs in other situations such as in optimization problems and in automatic curve fitting on the analog computer.

Curve storage on the analog computer imposes stringent equipment requirements. In the case of the serial solution of a parabolic partial differential equation approximately eighty percent of the amplifiers available in the computer are used for implementing curve storage and read out. A block diagram for the general technique for function storage on the analog computer is given in Figure 4-1.

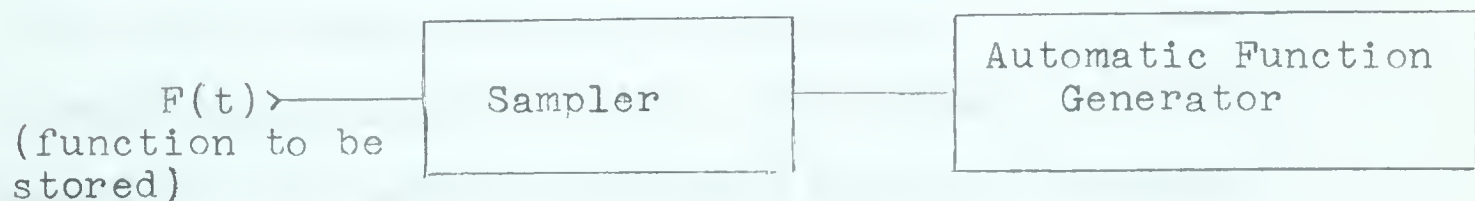


FIGURE 4-1. BLOCK DIAGRAM FOR FUNCTION STORAGE.

The sampler consists of a series of comparator controlled x-memories. One of these x-memories is given in Figure 4-2.



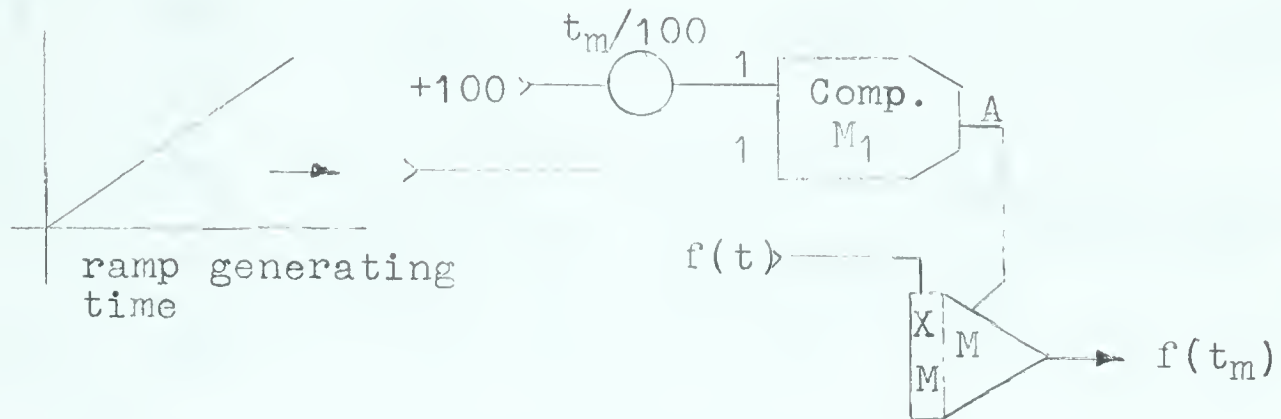


FIGURE 4-2. X-MEMORY IN SAMPLER

X-Memory M will track the input voltage  $f(t)$  until such time that  $t = t_m$ , when the comparator will switch the mode of M and the x-memory will now hold or store the value of  $f(t_m)$ . In a similar fashion other x-memories will store the values  $f(t_0)$ ,  $f(t_1)$ ,  $f(t_2)$ , ...,  $f(t_n)$ . In all further analysis the function will be considered to be sampled at equal intervals of time.

The automatic function generator will consist of a program on the analog computer such that using the stored values,  $f(t_0)$ ,  $f(t_1)$ , ...,  $f(t_n)$ , the continuous function  $f(t)$  can be regenerated when the program is driven by a voltage proportional to time. The automatic function generator will employ the Newton Forward Difference Interpolation formula for regeneration of the required function. The Newton Forward Difference Formula is;

$$f(t) = a + bt + \frac{ct(t-1)}{2!} + \frac{dt(t-1)(t-2)}{3!} + \dots$$

Coefficients a, b, c, and d are obtained in Table 4-1 (reference 1).



<u>t</u>	<u>f(t)</u>	<u><math>\Delta</math></u>	<u><math>\Delta^2</math></u>	<u><math>\Delta^3</math></u>
0	$K_0$			
1	$K_1$	$K_1 - K_0$	$K_2 - 2K_1 + K_0$	$K_3 - 3K_2 + 3K_1 - K_0$
2	$K_2$	$K_2 - K_1$	$K_3 - 2K_2 + K_1$	
3	$K_3$	$K_3 - K_2$	.	

TABLE 4-1. COEFFICIENTS FOR NEWTON'S  
FORWARD DIFFERENCE FORMULA.

From Table 4-1, it can be deduced that,

$$a = K_0$$

$$b = K_1 - K_0$$

$$c = K_2 - 2K_1 + K_0$$

$$d = K_3 - 3K_2 + 3K_1 - K_0$$

This procedure for finding the coefficients can easily be generalized to the case where more points are being used (See Appendix G).

#### 4.1 PROGRAMMING FOR AN AUTOMATIC FUNCTION GENERATOR

There are three schemes which can be used in programming an automatic function generator. A brief description of each will be given.

##### 4.1.1 METHOD 1. (DIRECT SIMULATION)

The scheme shown in Figure 4-3 is that for three stored points (reference 5). The generalization to more points



is immediate.

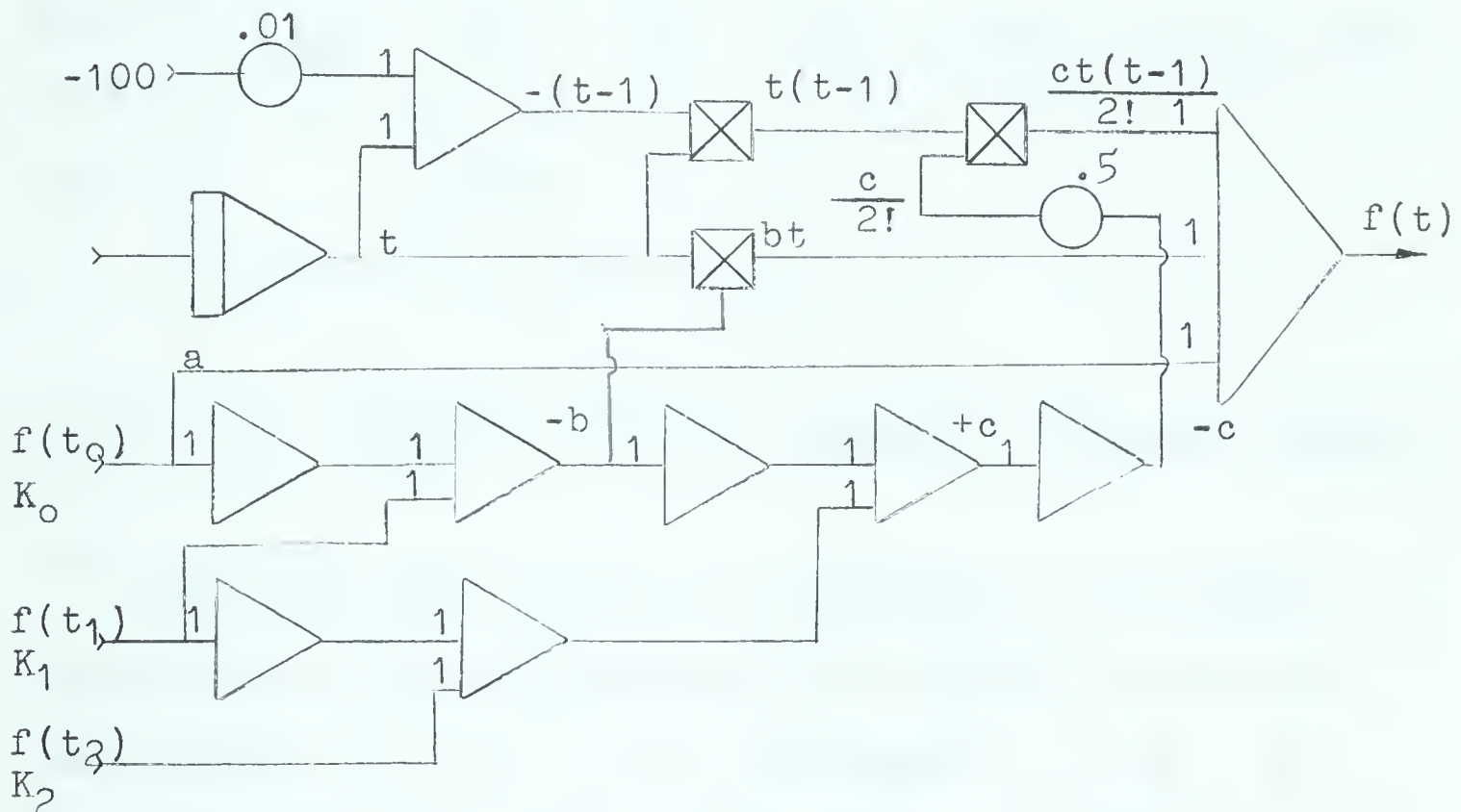


FIGURE 4-3. AUTOMATIC FUNCTION GENERATOR (DIRECT SIMULATION).

Using the above scheme for nine points would require sixty amplifiers and fifteen multipliers. As is evident, this scheme could be used successfully for up to about four points. However if better accuracy is required and hence more points must be used in the curve extrapolation, the excessive requirement for multipliers makes this scheme impractical.

#### 4.1.2 METHOD 2. (INDIRECT SIMULATION)

A block diagram of the scheme employed is given in Figure 4-4 (reference 7).



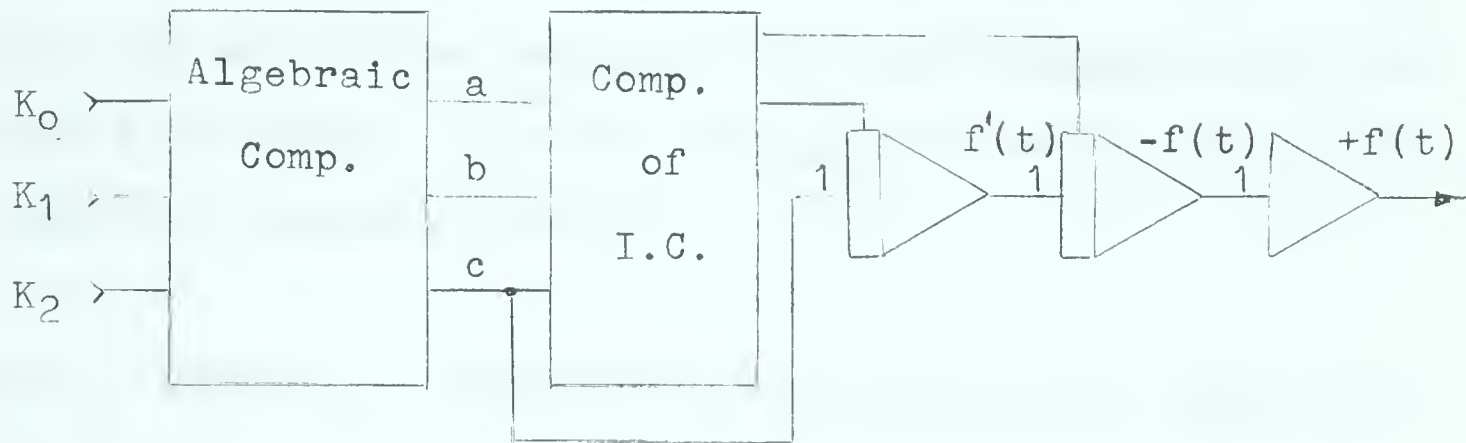


FIGURE 4-4. AUTOMATIC FUNCTION GENERATOR (INDIRECT METHOD).

Using the table drawn up in Appendix G, an algebraic calculation can be performed on the computer to find the coefficients  $a, b, c, \dots, n$ . For example,  $b = K_1 - K_0$ . If the function and its derivatives are considered at zero we have;

$$\begin{aligned} f(t) &= a + bt + \frac{ct(t-1)}{2!} + \dots \\ f(0) &= a \\ f'(t) &= b + ct \\ f'(0) &= b \\ f''(t) &= c \\ f''(0) &= c \end{aligned}$$

After all the algebraic calculations for the values  $a, b, c$  are completed, a second algebraic calculation can be performed to give the value of the function as well as all of its derivatives at  $t = 0$ . Examining the equation more carefully shows that the highest non-zero derivative of the polynomial gives the derivative of the function throughout the entire



period of interest. In the above case,  $f''(t) = c$ . Since  $f''(t)$  and all initial values of the derivatives of all lower orders are known, successive integration may be applied to obtain the desired function.

#### 4.1.3 METHOD 3. (NEWTON'S ITERATIVE FUNCTION GENERATOR)

The technique employed here is to use two sets of Newton's Function Generators, each programmed for  $n$  points using the technique described in Method 2. Each is switched in or out of operation by a comparator, hence giving a total possible number of  $2n$  stored points. A block diagram is given in Figure 4-5.

This technique can easily be generalized to provide a means for extrapolating stored data from a digital memory. The stored function could then be read out in the form of a polynomial rather than in box-car form. This is depicted graphically in Figure 4-5a.

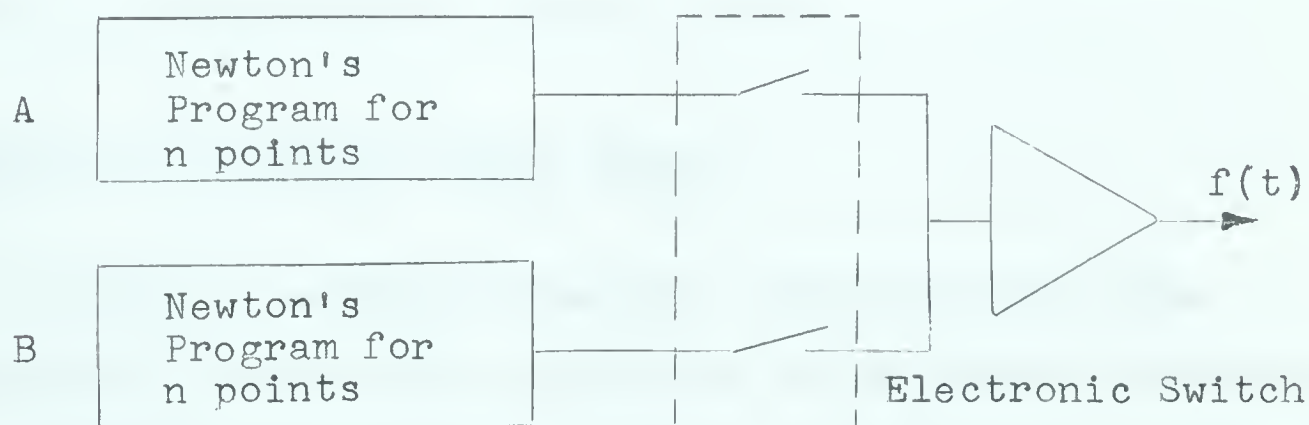


FIGURE 4-5. BLOCK DIAGRAM FOR NEWTON'S ITERATIVE FUNCTION GENERATOR.



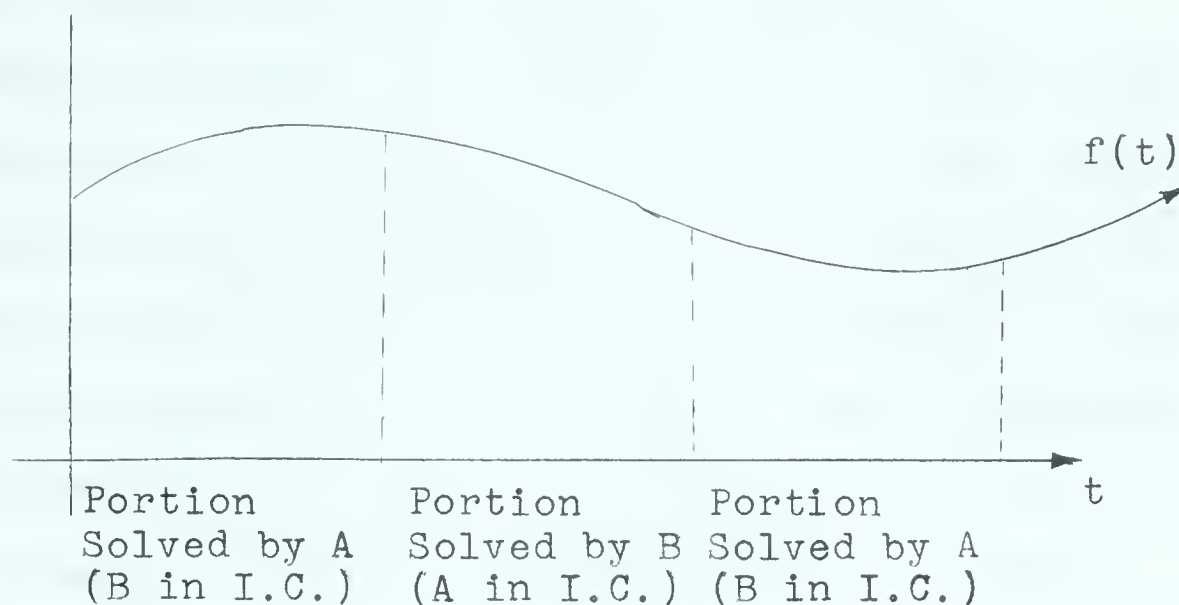


FIGURE 4-5a. NEWTON'S ITERATIVE FUNCTION GENERATOR  
FOR DIGITAL TO ANALOG CONVERTER

A brief description of the operation of Newton's Iterative Function Generator as a digital to analog converter is given with reference to Figure 4-5. When program A is in the operate mode, B is in the initial condition mode with its output switch open. At the end of operate cycle of program A, the output switch of A will open, that of B will close and B will now be in the operate mode. This sequence is then repeated to extrapolate the entire curve.

#### 4.2 NEWTON'S PROGRAM FOR NINE POINTS

To program an automatic function generator for nine points requires twenty-five amplifiers using Newton's Iterative Method as compared to sixty amplifiers and fifteen multipliers for Method 1, and seventy amplifiers for Method 2. The



advantage of employing this technique is evident.

A memory programmed for 9 points will be used in the serial solution of a parabolic partial differential equation. Two automatic function generators, each of five points are programmed and are switched in the final integrator, that is, the integrator generating  $-f(t)$ . This is done because an attempt to perform switching in the summing junction of an amplifier tends to produce a ringing effect. A block diagram for an automatic function generator using 9 stored points is given in Figure 4-6.

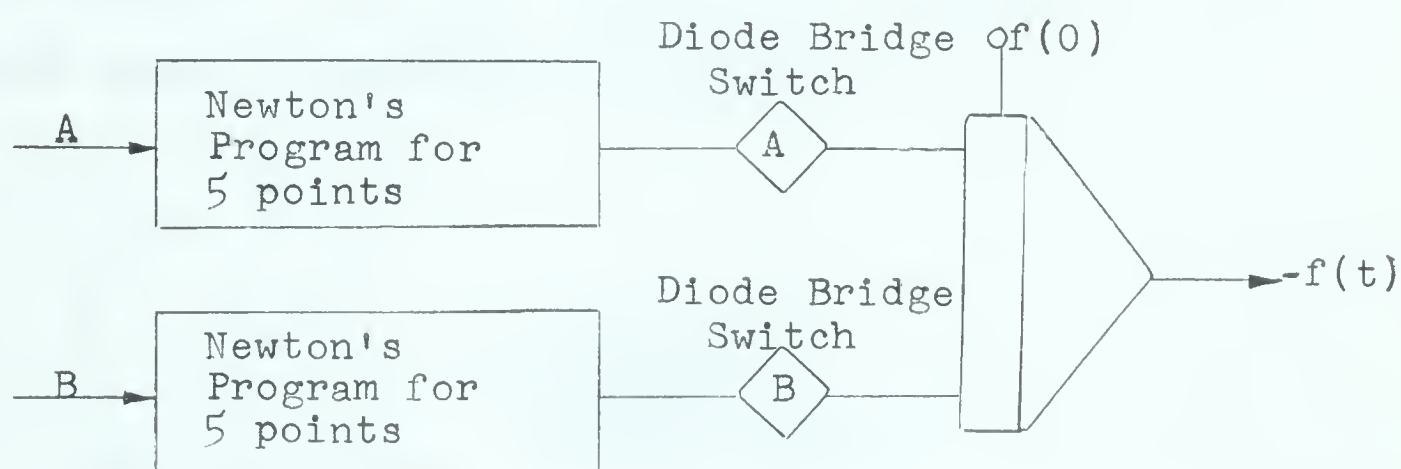


FIGURE 4-6. FUNCTION GENERATOR BLOCK DIAGRAM (9 POINTS).

The method of operation of the program in Figure 4-6 is similar to that described for the Newton's Iterative Function Generator. However the initial conditions are read in simultaneously to both programs when the computer is in the reset mode. Program A then generates the function for the first four segments during which time program B is forced to remain in the initial condition mode (with switch B open).



At  $x = L/2$ , switch A is opened and switch B is closed. Program B now generates the remaining half of the function. No extra initial condition is required for B since the desired value or initial condition for program B is already existing on integrator 1 when A completes its operate cycle. Program A is forced to remain in the hold mode. Otherwise program A would continue the generation of some extension of the first half of the desired curve and eventually saturate some amplifiers. The technique employed for having B remain in the I.C. mode for the first half of the operate cycle and to have A remain in the hold mode for the remainder of the operate cycle is described using Figure 4-7.

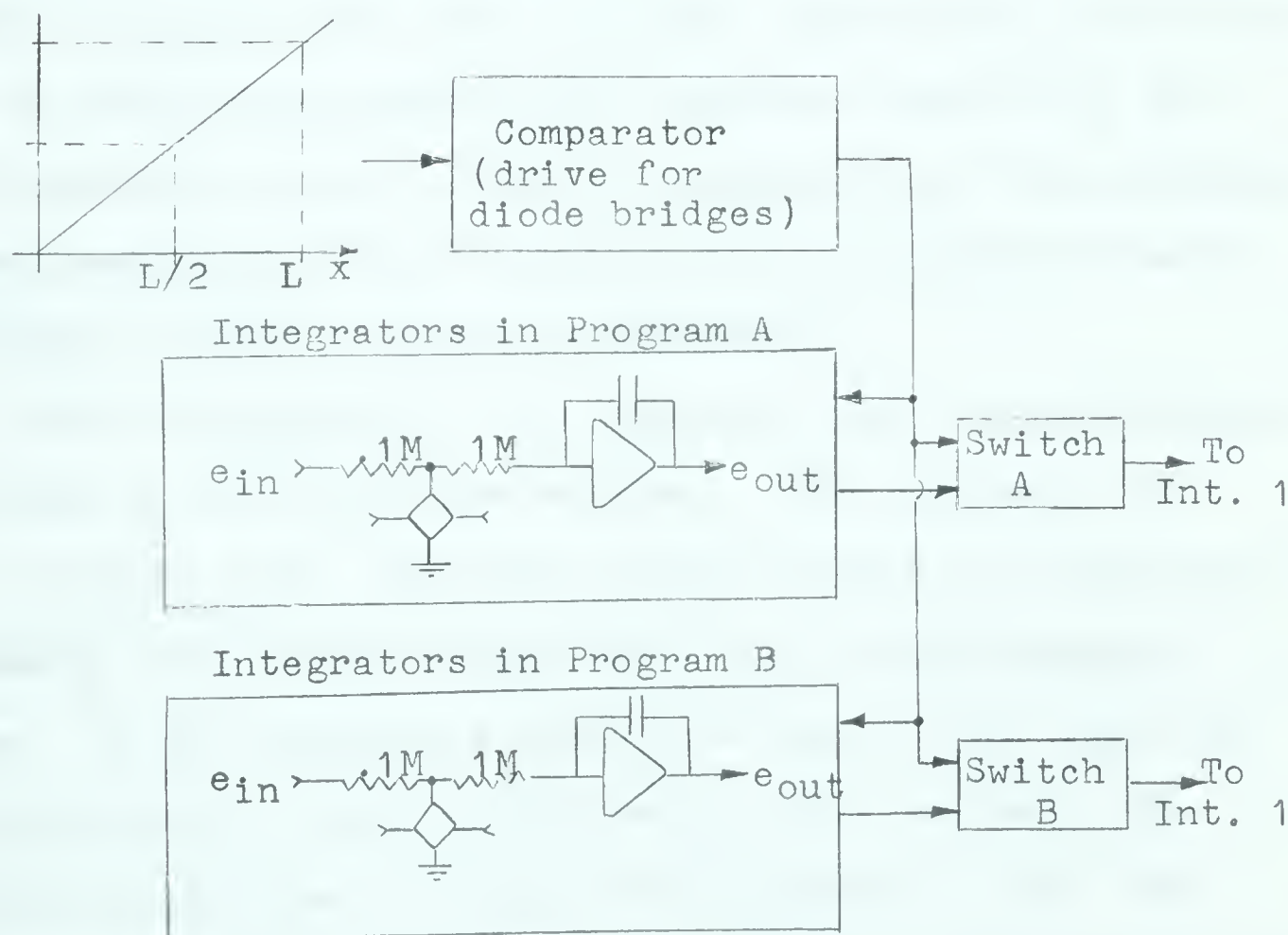


FIGURE 4-7. MODE CONTROL FOR FUNCTION GENERATOR.



In the first portion of the operate cycle diode bridges in Program A remain non-conducting. Hence the integrators will operate in the normal fashion until  $x = L/2$ . In program B all bridges are conducting, forcing the inputs to all integrators to zero. These integrators will remain holding their initial conditions until  $x = L/2$ . At this instant, the comparator will switch polarities of the control voltages to the bridges, causing those in program A to conduct and in turn forcing the input to each of these integrators to zero. As a result integrators in Program A will hold their value from  $x = L/2$ . Switch A is open, hence A will have no effect on the remainder of the solution. Diode bridges on integrators in program B will now be non-conducting and hence program B will now be in the operate mode, and with switch B closed will generate the remaining portion of the stored function. A more complete description of the operation and performance of the mode control of the integrators used in programs A and B is given in Appendix F.

A brief description of the operation and the performance of switches A and B will now be given. The schematic is given in Figure 4-8. Initially, when bridge A is conducting, the summing junction of integrator 1 is at approximately 0 volts. If the protective diodes are used at the input to the diode switch, a small undesirable step occurs at the switching instant in the output of integrator 1. If however, the diodes are left out, the circuit is still protected by the clamping diodes in the flip-flops driving the bridge.



In this way, switching by A is almost ideal.

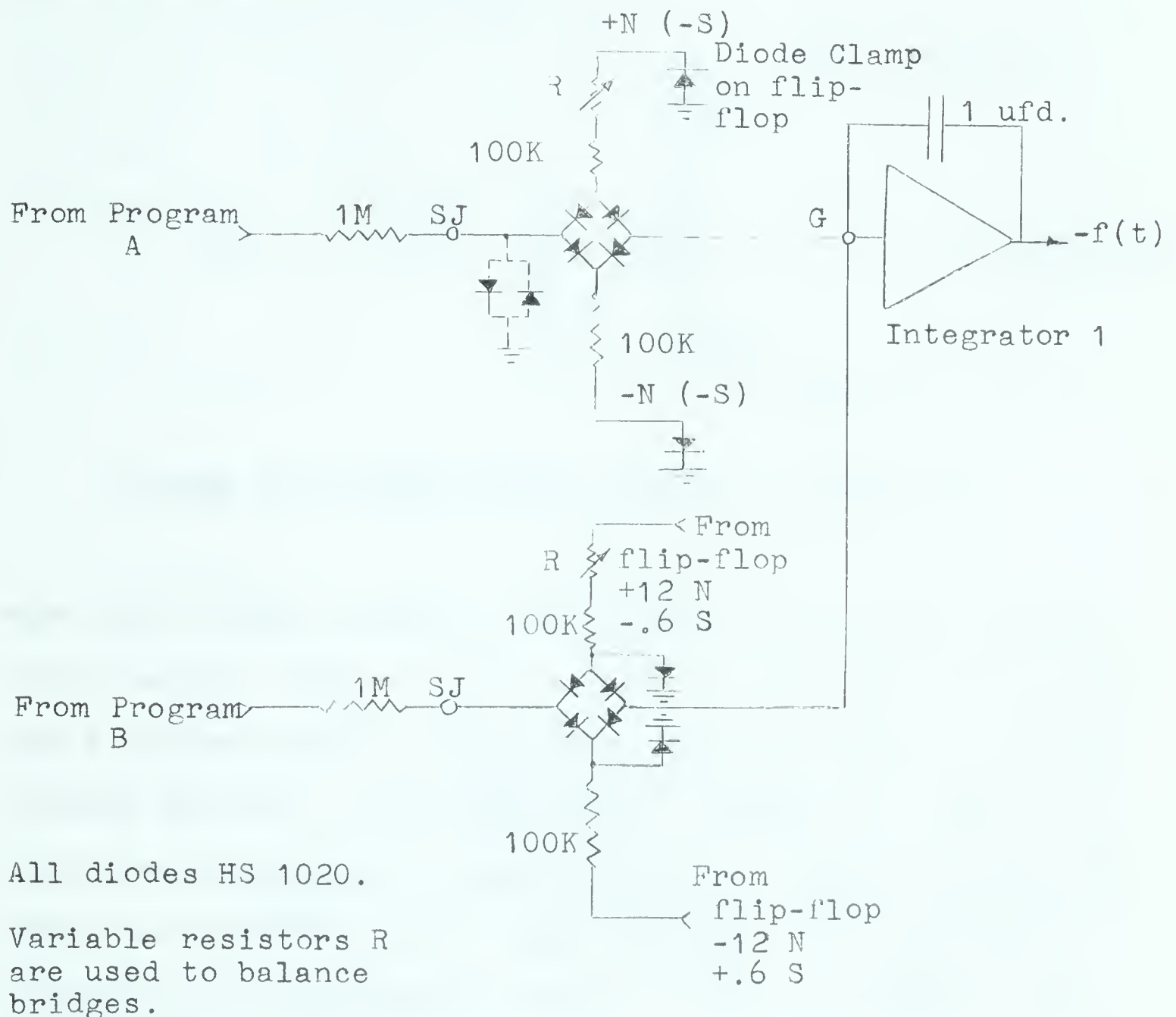


FIGURE 4-8. SWITCHES A AND B

Diode bridge B is cut-off in the first portion of the operate cycle. To prevent the input side of the diode bridge from rising to a large voltage (to the input voltage), protective diodes are used at the input to the bridge. However, at the instant the bridge starts to conduct, a small undesirable step occurs in the output of integrator 1.



The scheme used for placing a diode bridge switch in an integrator is given in Figure 4-9.

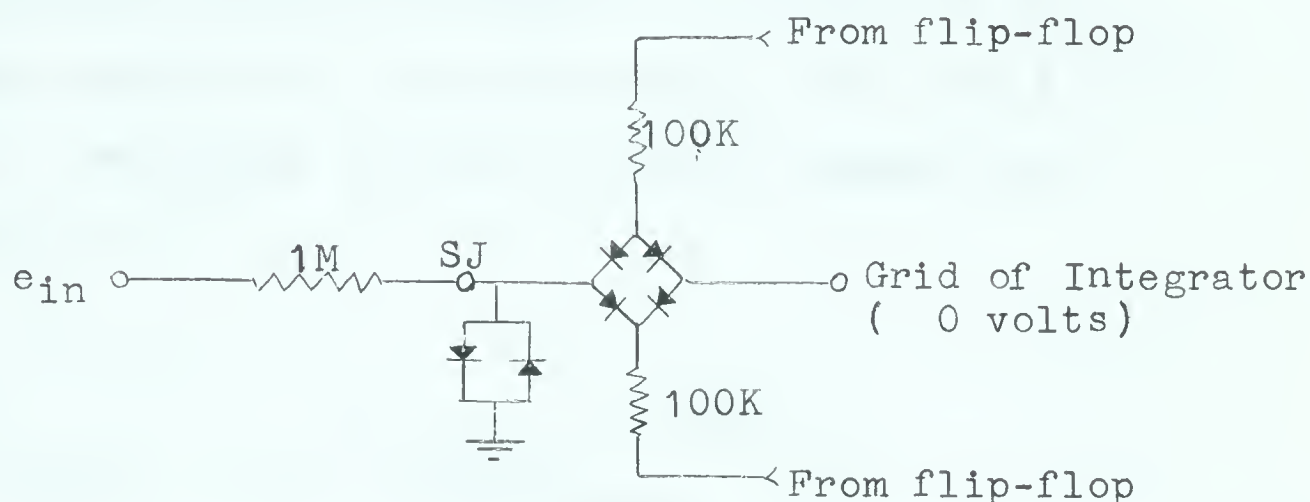


FIGURE 4-9. DIODE BRIDGE SWITCH IN INTEGRATOR.

When the bridge is cut off, the diodes have a small charge stored across them. At the instant the bridge switches, the small charge stored in the diodes will give rise to a small voltage directly to the grid of the integrator. This will give rise to a step of approximately 0.3 volts, the sign of which is dependent on the input voltage. The equivalent circuit at the switching instant is given in Figure 4-10.

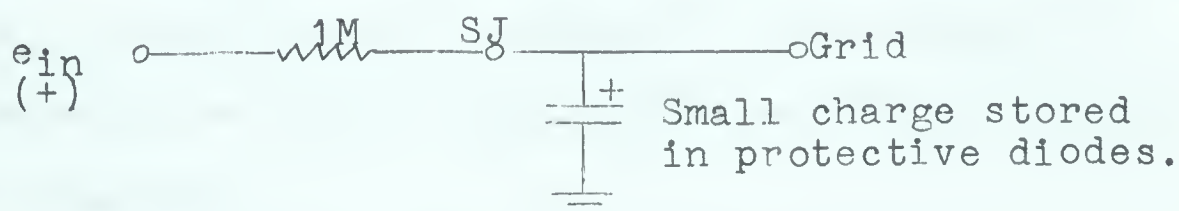


FIGURE 4-10. EQUIVALENT CIRCUIT FOR DIODE BRIDGE SWITCH.

An improvement by a factor of two to three was obtained using the circuit shown in Figure 4-8.



### 4.3 FUNCTION STORAGE USING NINE POINTS

The program for the storage of a function of the independent variable  $x$ , using nine points will now be developed. The scheme is given in block diagram form in Figure 4-11.

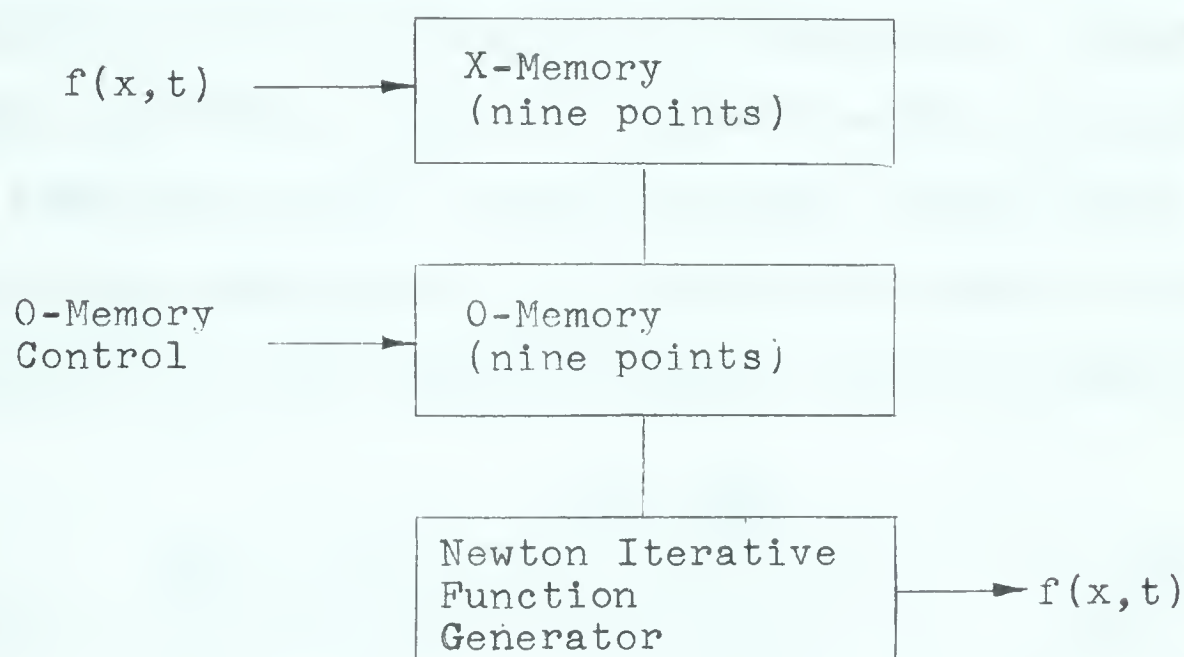


FIGURE 4-11. BLOCK DIAGRAM FOR FUNCTION STORAGE.

#### 4.3.1 THE X-MEMORY

The X-Memory is analogous to the sampler previously described. It consists of a series of nine comparator controlled memories which store the value of the function at equal intervals.

The computer scheme for the input to the X-Memory is given in Figure 4-12.



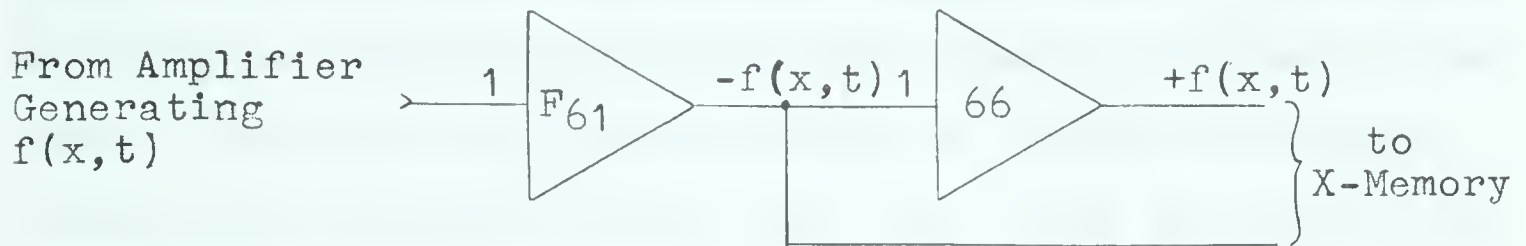


FIGURE 4-12. INPUT TO X-MEMORY.

The computer scheme for the Memory storing the value of the function at  $x = L$ , is given in Figure 4-13. Integrator 10 simply generates a ramp, the value of which is proportional to  $x$ . Amplifier 44 will track the input voltage until  $x = L$ , at which time comparator 11 will switch the mode of 44 to store. However at  $x = L$ , the computer operate cycle is completed and

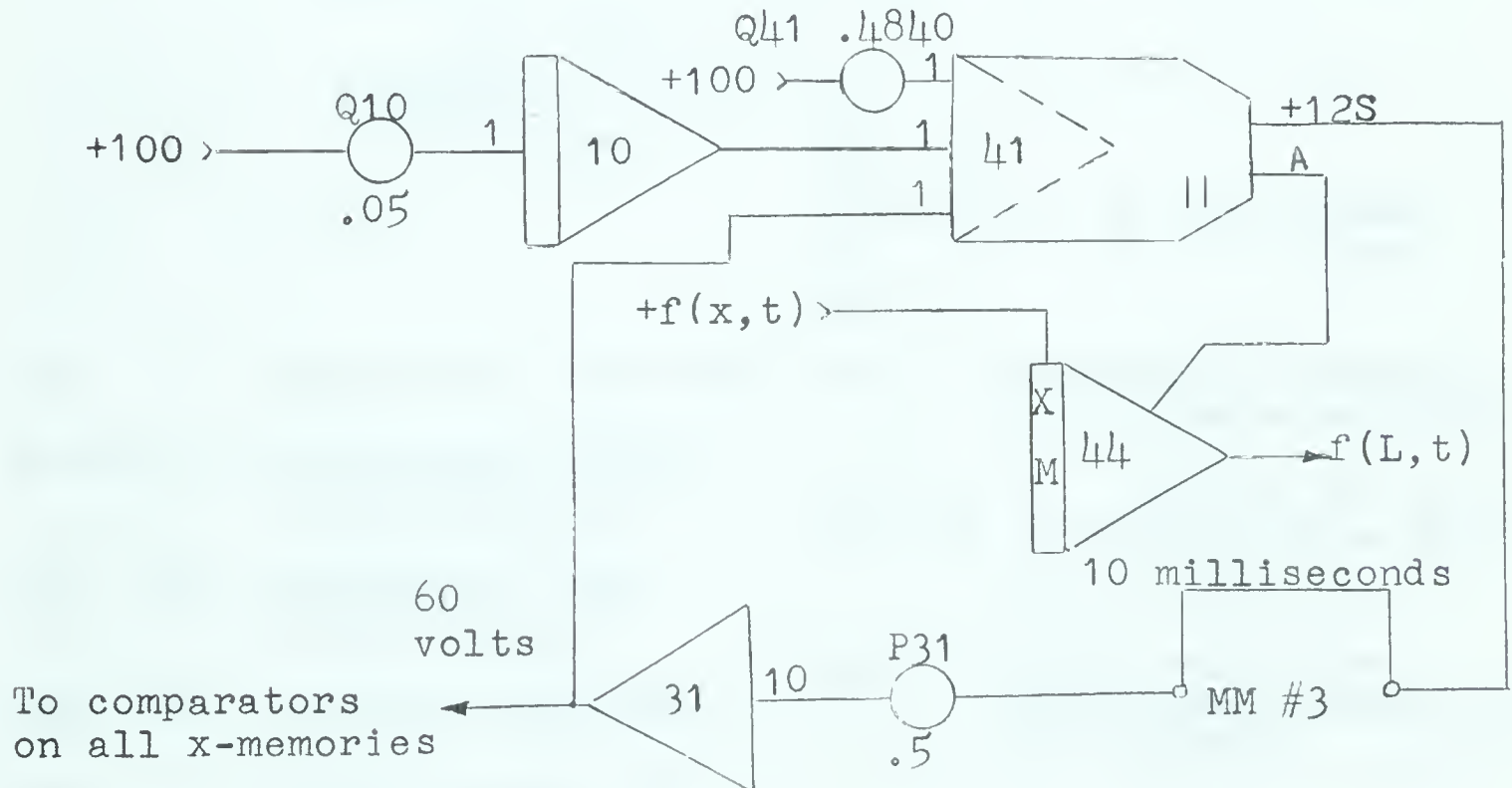


FIGURE 4-13. X-MEMORY STORING  $f(L)$ .

the computer will go into the reset mode. This would mean that comparator 11 would immediately switch back to the normal



state, causing the X-Memory 44 to again track. This would not provide sufficient time for the O-Memory to read in the data. For this reason MM#3 is used to provide an inhibit or latching voltage to insure that the value in each of the X-Memories is held for 10 milliseconds after the operate cycle is completed.

The symbols used in Table 4-2 listing the components of the X-Memory are explained in Figure 4-14.

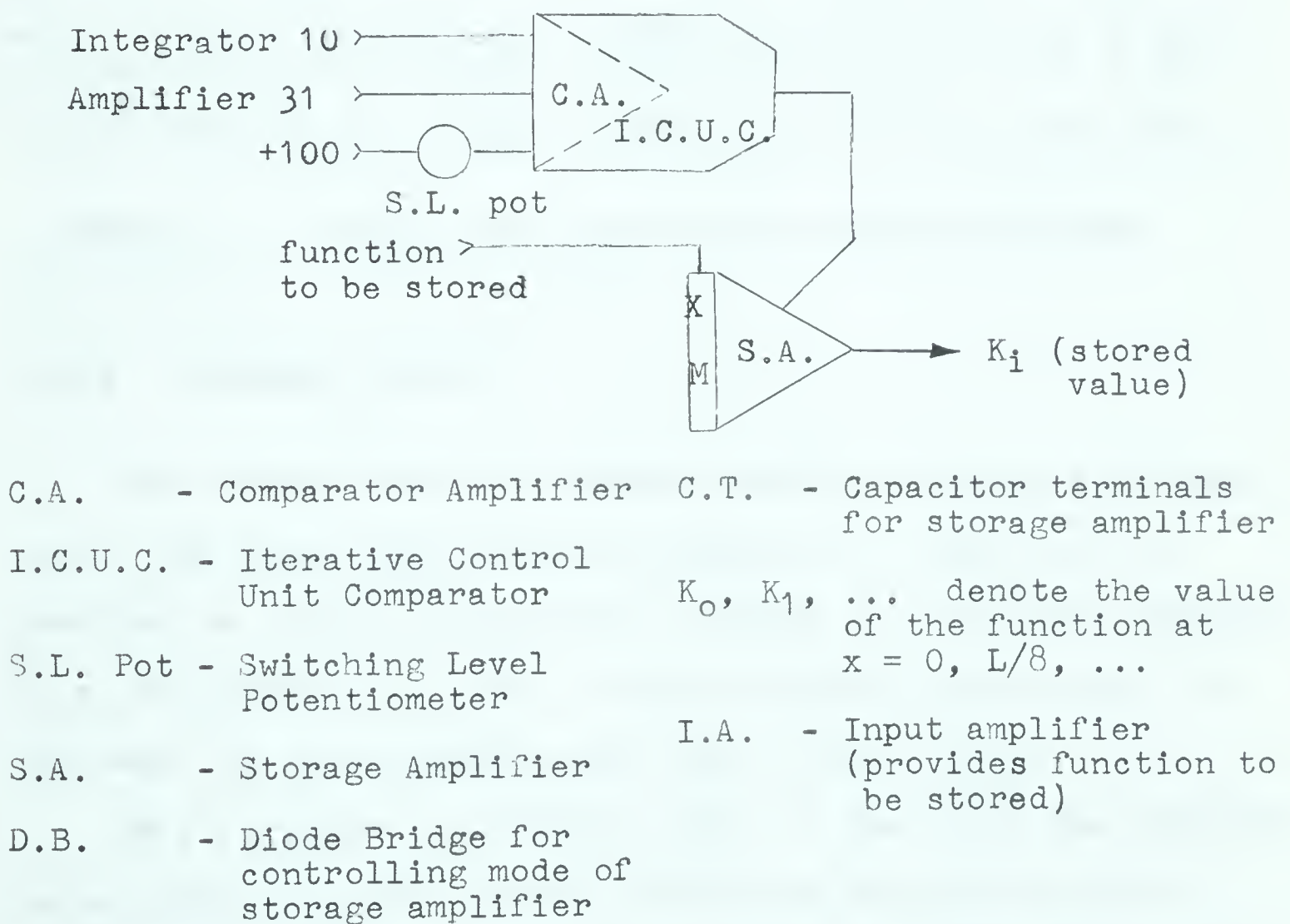


FIGURE 4-14. X-MEMORY SYMBOLS



Value Stored	S.A.	ICUC	S.L. Pot	S.L. Pot S.	D.B.	C.T.	C.A.	I.A.
K <sub>0</sub>	47	1	Q15	.0915	16	B06 B07	15	I.C.
- K <sub>1</sub>	48	4	Q16	.0450	17	B08 B09	16	F <sub>61</sub>
+ K <sub>2</sub>	49	5	Q25	.1110	18	C06 C07	25	66
- K <sub>3</sub>	32	6	Q26	.1750	19	C08 C09	26	F <sub>61</sub>
+ K <sub>4</sub>	33	7	Q35	.2375	30	D06 D07	35	66
- K <sub>5</sub>	34	8	Q36	.2980	21	D08 D09	36	F <sub>61</sub>
+ K <sub>6</sub>	42	9	Q45	.3642	22	10 110	45	66
- K <sub>7</sub>	43	10	Q46	.4250	23	11 111	46	F <sub>61</sub>
+ K <sub>8</sub>	44	11	Q41	.4840	29	12 112	41	66

TABLE 4-2. X-MEMORY FOR NINE POINT FUNCTION STORAGE.

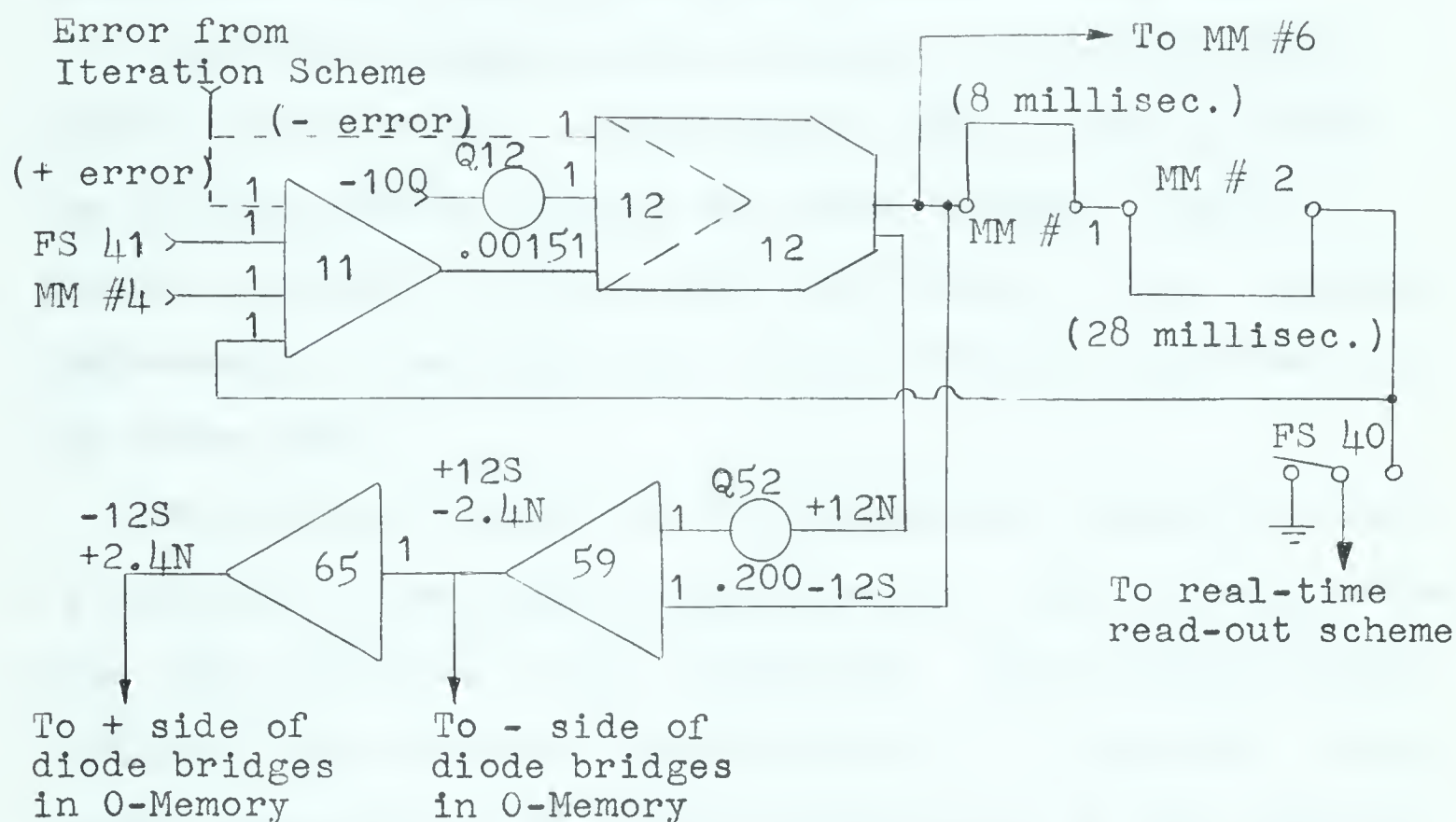
#### 4.3.2 O-MEMORY CONTROL

The scheme used for O-Memory control is given in Figure 4-15. The input from function switch 41 to amplifier 11 provides an inhibit pulse when a scheme for obtaining results on an x-y plotter is used. A more complete explanation for the need for this inhibit pulse will follow later.

MM #4 provides an inhibit pulse so that when the computer is switched from the "Initial Condition Read-In position" to "operate", in the serial solution of a partial differential equation, at least one iteration for the solution of the next interval will take place. This will insure that after the



switch from I.C. to operate is made, that the comparator does not switch during the next operate cycle causing the information stored in the O-Memory to be destroyed.





O-Memory will hold the data just read in for the remainder of the reset cycle as well as for the next operate cycle. Otherwise comparator 12 would remain in the switched state during the reset cycle as well as the next operate cycle, causing the data read in from the X-Memory to be lost.

All diode bridges in the O-Memory are switched from amplifiers 59 and 65. Potentiometer Q52 is used to limit the voltage used to cut-off the diode bridges. Using a smaller voltage to reverse-bias the bridges causes a marked improvement in reducing drift of the O-Memory amplifiers in the store mode.

By reading in data for 8 milliseconds, sufficient time is provided to use larger capacitors, .01 ufd, in conjunction with 100K resistors in the O-Memories. Error during read-in employing these values is approximately 0.03 percent. Using larger capacitors in the storage amplifiers of the O-Memory also makes the real time plotting scheme more feasible, where the value in the O-Memory must be held for at least seven seconds.

#### 4.3.3 O-MEMORY

The O-Memory consists of nine memories controlled from Comparator 12 as described in the previous section. A small bias directly to the grid of each of the storing amplifiers helps to overcome drift due to unbalance in the diode bridges. After sufficient time has been allowed for computer warm up,



drift in the O-Memory can be limited to less than .02 volts for 10 seconds using this technique. Both negative and positive compensation has been provided. The schematic for a typical O-Memory is given in Figure 4-16.

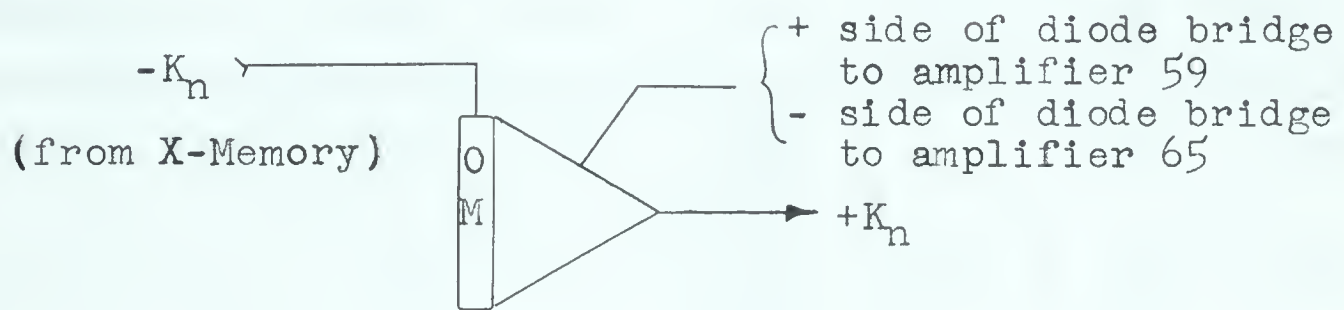


FIGURE 4-16. TYPICAL O-MEMORY

The compensation for bridge unbalance is given in Figure 4-17.

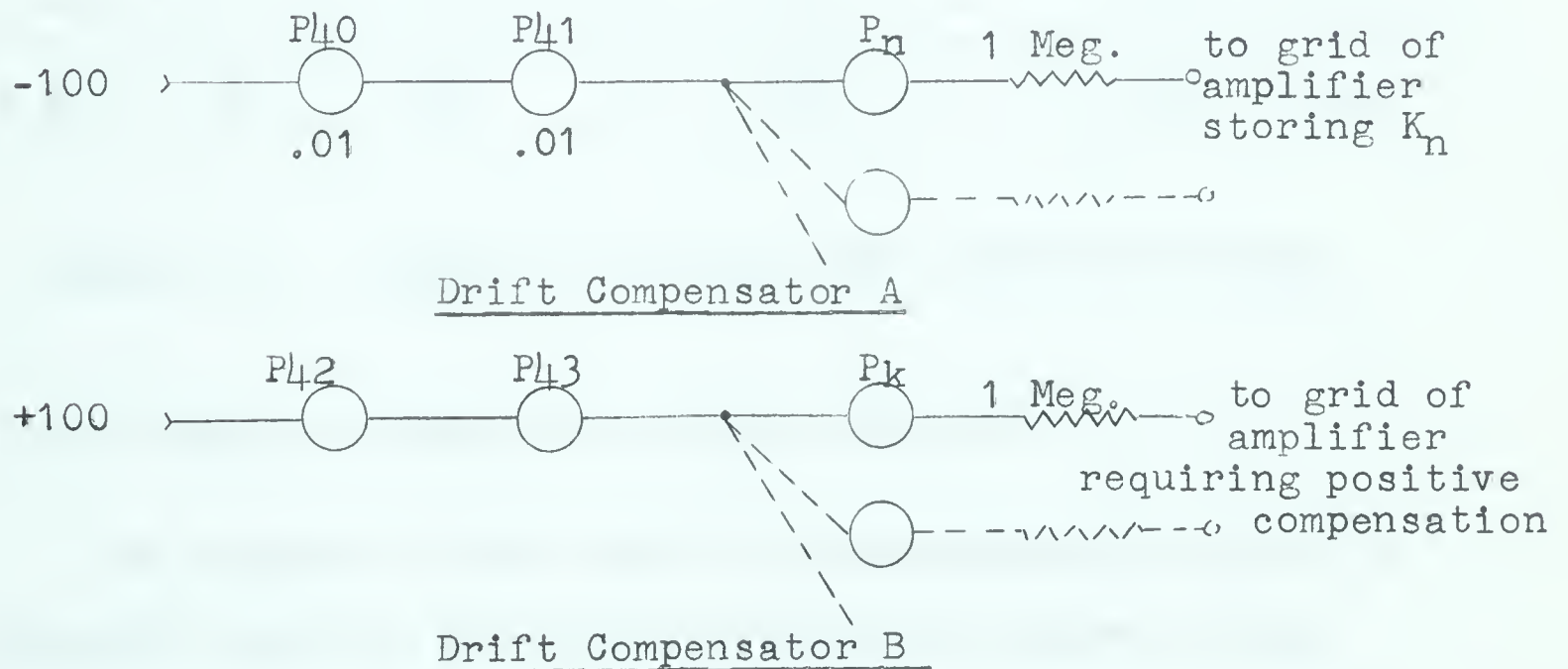


FIGURE 4-17. COMPENSATION FOR DRIFT IN O-MEMORY.



Table 4-3 summarizes the amplifiers used, the diode bridges, capacitor terminals, as well as the required drift compensation for the O-Memory. For explanation of symbols, see Figure 4-14.

Value Stored	S.A.	I.A.	D.B.	C.T.	Drift Compensation		
					Comp.	Pot.	Pot. Setting
+ K <sub>0</sub>	17	47	7	00 100	A	P17	.0865
- K <sub>1</sub>	18	48	8	01 101	B	P18	.1230
+ K <sub>2</sub>	19	49	9	02 102	A	P19	.1390
- K <sub>3</sub>	27	32	10	03 103	A	P27	.1396
+ K <sub>4</sub>	28	33	11	04 104	B	P28	.1200
- K <sub>5</sub>	29	34	12	05 105	A	P29	.3380
+ K <sub>6</sub>	37	42	13	06 106	B	P37	.1654
- K <sub>7</sub>	38	43	14	07 107	A	P38	.0295
+ K <sub>8</sub>	39	44	15	08 108	A	P39	.0547

TABLE 4-3. O-MEMORY FOR NINE POINT FUNCTION STORAGE.

#### 4.3.4 NEWTON'S ITERATIVE FUNCTION GENERATOR

The computer scheme used for programming the Newton's Iterative Function Generator is given in Figures 4-18a, 4-18b, 4-18c, and 4-18d. Integrator 91 generates a ramp which is proportional to x. Comparator #2 performs the switching functions described by Switches A and B while



comparator #3 performs the function of holding program B in initial condition for the first portion of the operate cycle, then switching program A into the hold mode for the remaining half of the operate cycle. The arrangement of amplifiers 90 and 81 is necessary since the removal of the pot set relay from integrator 50 means that the initial condition on this integrator would normally be  $-K_0 - f'(0)$ . The portion of the computer designated with a bar above all quantities, for example  $\bar{e}$ , refers to program B of Newton's Iterative Function Generator. The complete computer program for the function generator is given in Figures 4-18a, 4-18b, 4-18c and 4-18d.



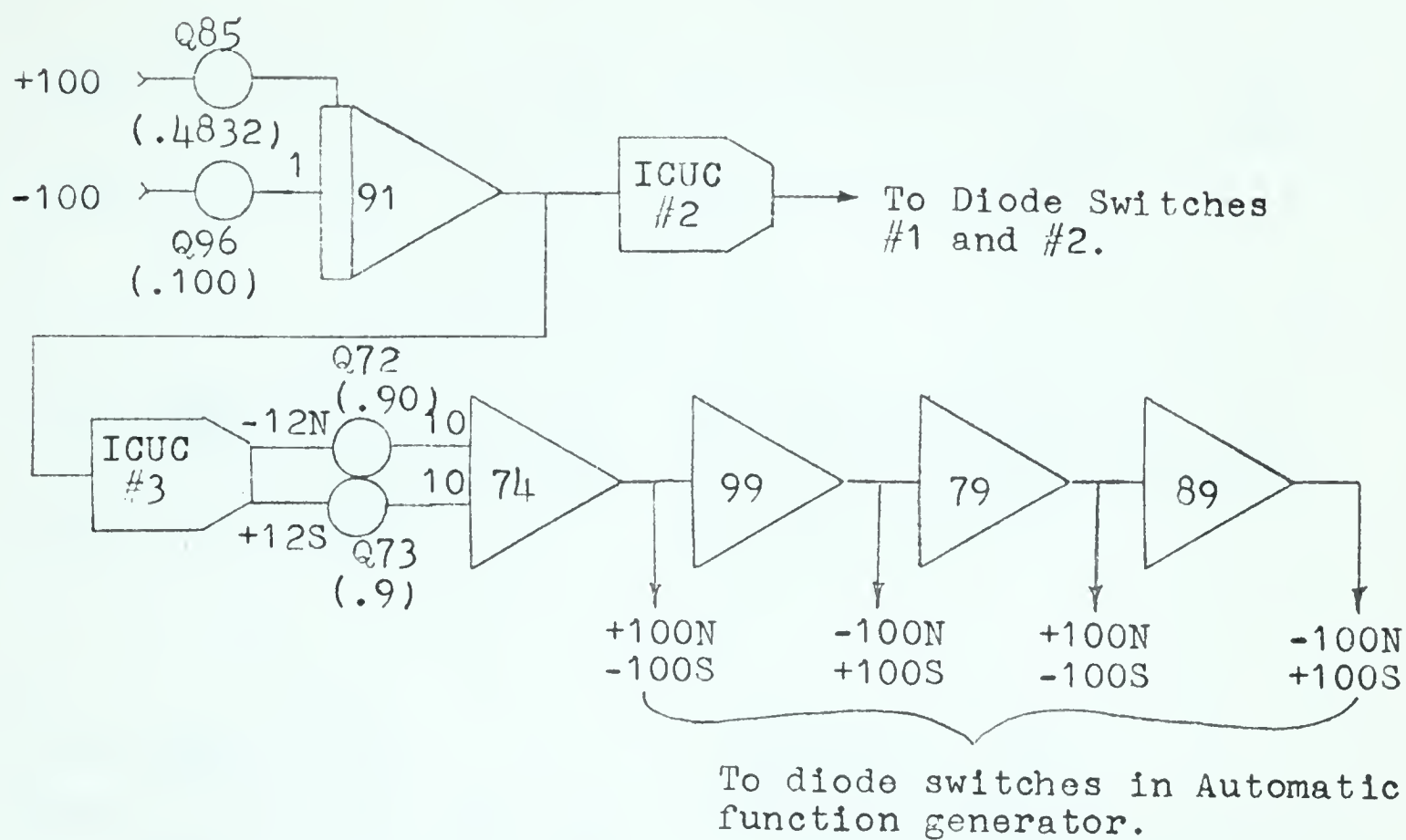


FIGURE 4-18a. SWITCHING FOR THE FUNCTION GENERATOR

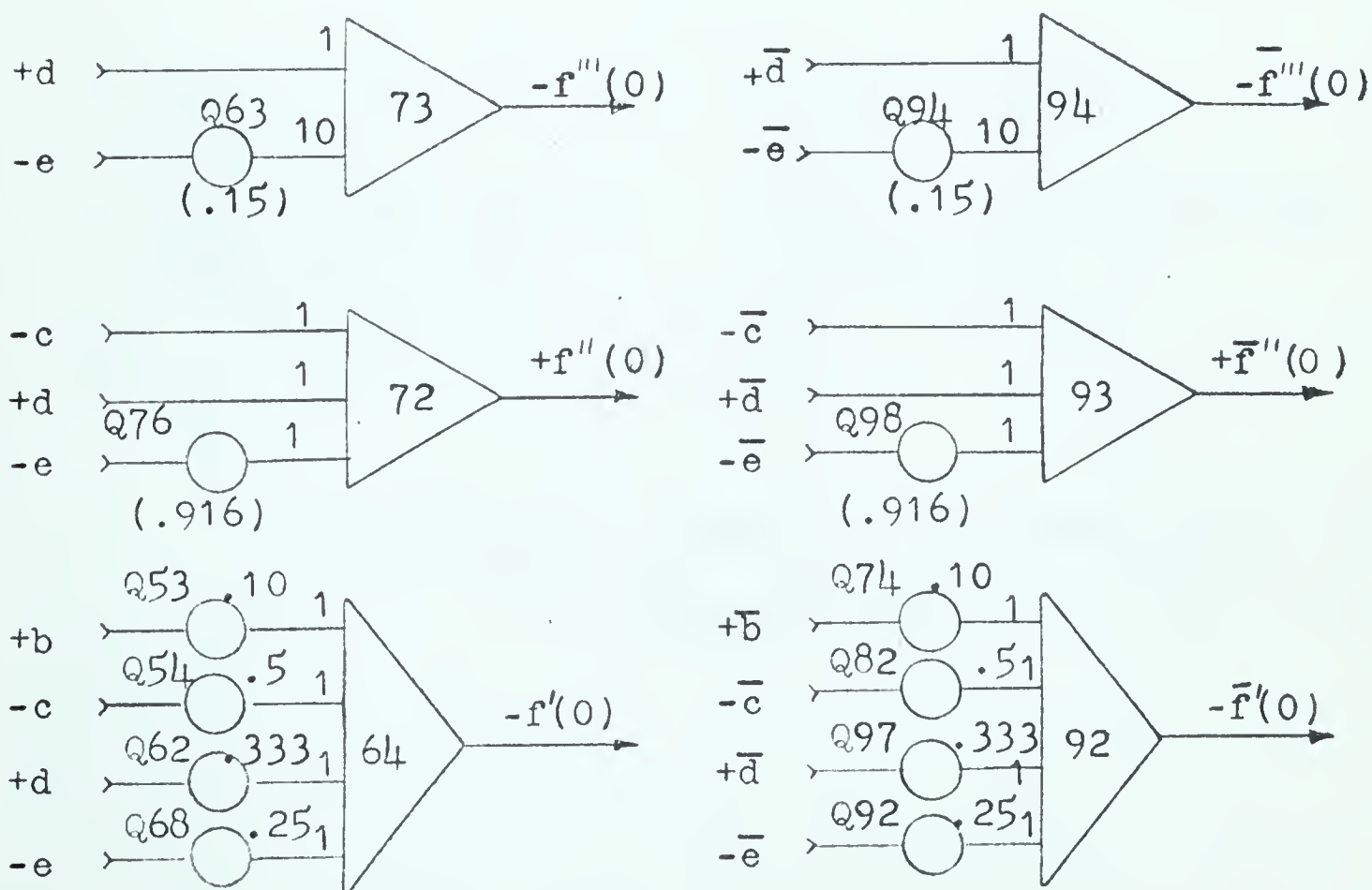


FIGURE 4-18b. ALGEBRAIC CALCULATIONS FOR I.C.



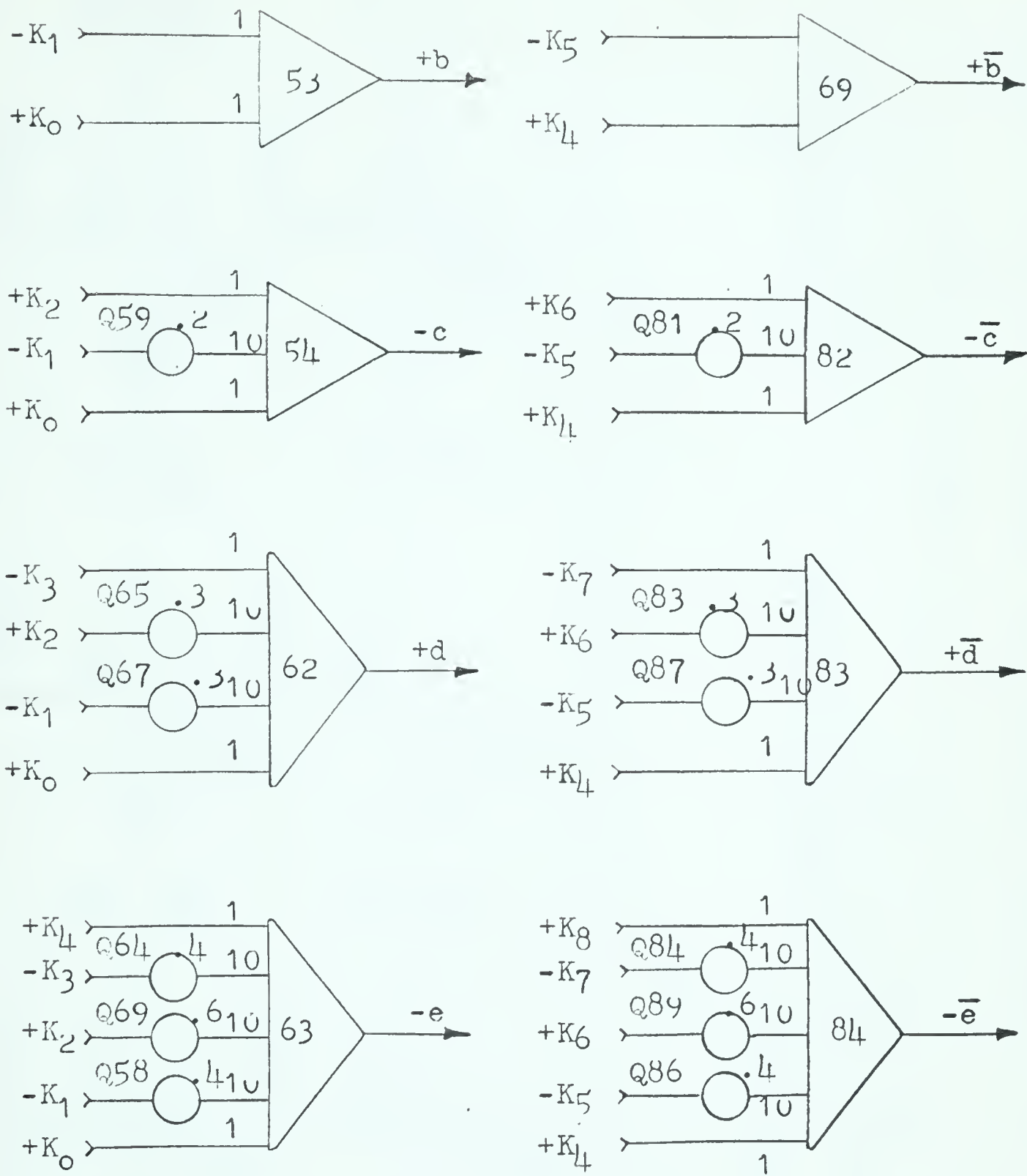


FIGURE 4-18c. ALGEBRAIC CALCULATIONS FOR  
COEFFICIENTS a to e.



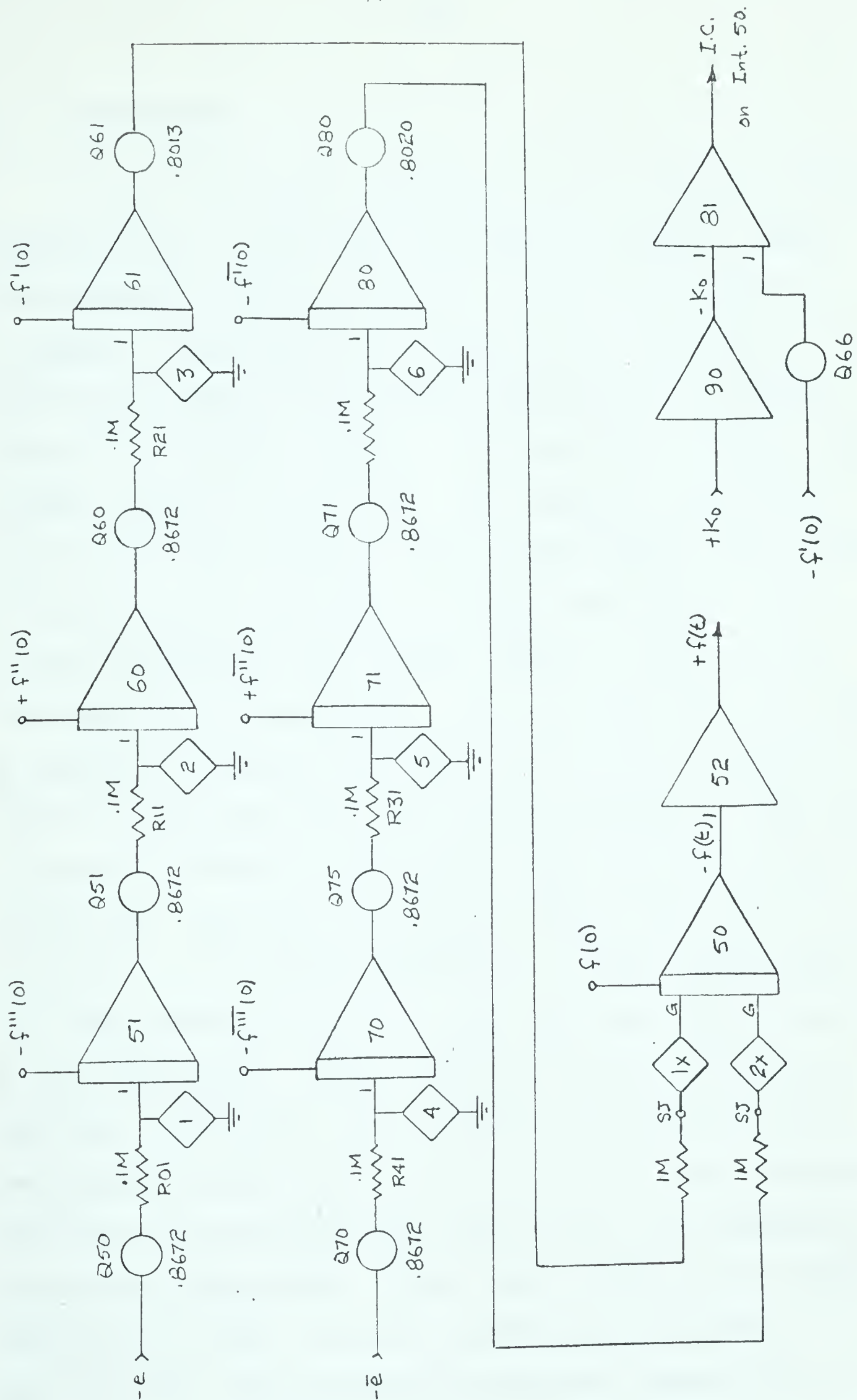


FIGURE 4-18d. COMPUTER PROGRAM FOR ITERATIVE FUNCTION GENERATOR.



#### 4.4 CONCLUSIONS

The use of Newton's Iterative Function Generator for extrapolating a stored curve provides a practical means for performing function storage on the analog computer. If digital storage for the function is not available, then two programs of five points represents the optimum choice for function storage. The reason that this is the optimum choice is as follows. The number of amplifiers required for two programs each of four points (total number of stored points is seven) is twenty amplifiers. That for two programs of five points, making a total of nine stored points is twenty-four. It is evident that the number of amplifiers required per point for the two programs of five points is less than that for the two programs of four points. A function generator consisting of two programs, each of five points also represents an optimum choice with respect to the accuracy of the function generator. In the calculations for the coefficients  $a$ ,  $b$ ,  $c$ ,  $\dots e$ , gains of less than 10 are used on all amplifiers. However gains of twenty are required on amplifiers performing these algebraic calculations for a seven point program and gains of thirty five are required on amplifiers calculating these coefficients in a program for 8 points (see Appendix G). This implies that if the stored function is a ramp, where all coefficients following  $b$  are equal to zero, the errors will accumulate rapidly in the algebraic calculation for the last coefficients.

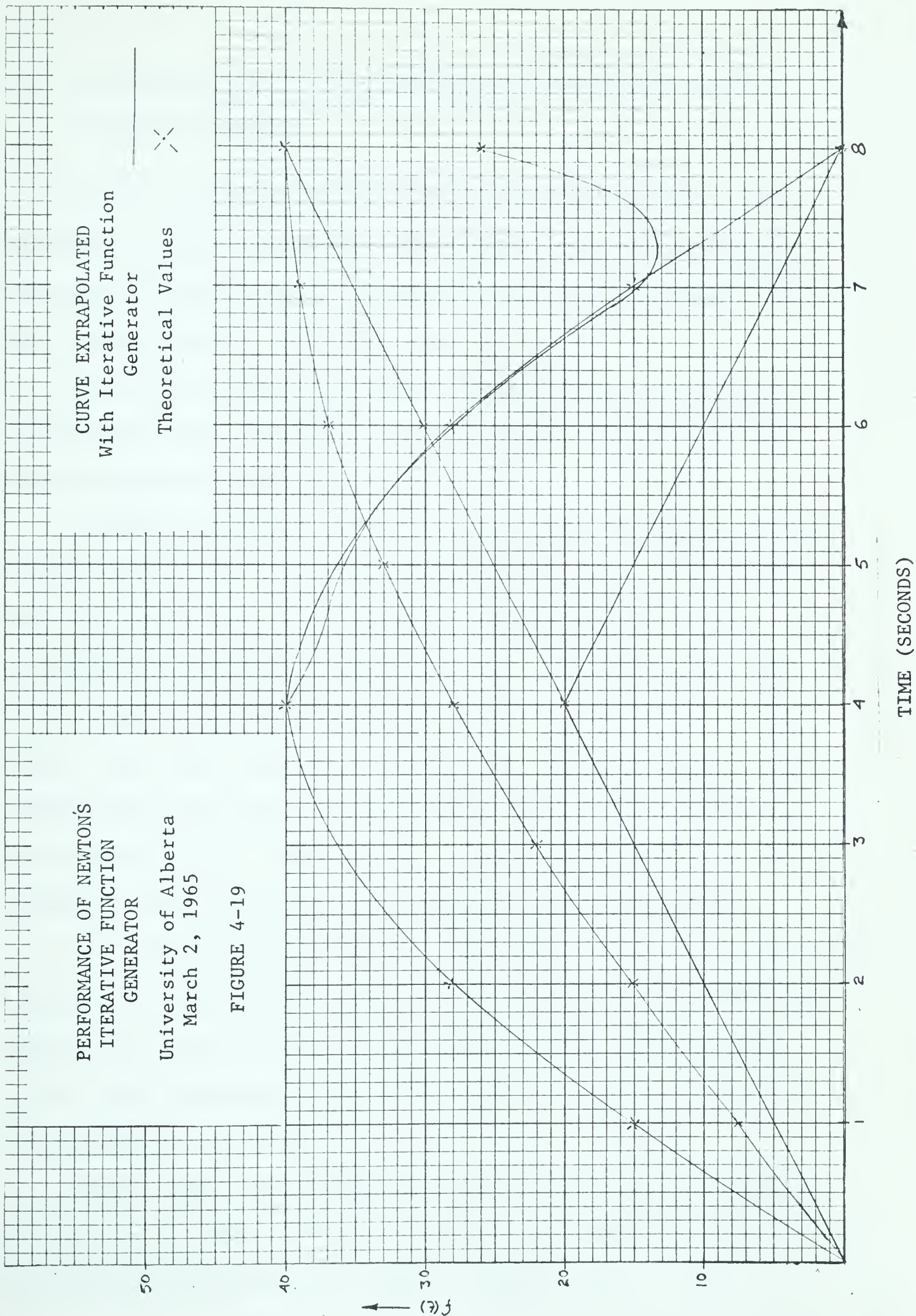


For example, in the algebraic calculation for  $i$ , the difference between eight quantities, each with large gains (up to 35) is to be obtained. This will tend to make the required potentiometer settings extremely critical.

Using field-effect transistors for implementing mode control, Newton's Iterative Method may be used for parabolic curve extrapolation (that is, each program consists of 3 points) using any number of stored points. The iterative function generator program however will only require fifteen amplifiers. This technique makes the implementation of function storage realizable even with smaller computers such as the Pace TR-48.

The accuracy of the extrapolation of the stored function performed by the iterative function generator is dependent upon the irregularity of the stored function. If the rate of change of the function is extreme, then more points will be required to represent the function more accurately. Using parabolic extrapolation with mode control on the integrators performed by field-effect transistors, this iterative function generator can be used to extrapolate functions with only piecewise continuous derivatives. The error of the extrapolated function at the instants the value of the function is stored is less than 0.2 percent. The performance of the iterative function generator for nine stored points is given in Figure 4-19.







## 5.0 SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS ON THE ANALOG COMPUTER

In this chapter the solution of partial differential equations in two independent variables will be considered. There are three general techniques that can be applied for the analog solution of these equations.

The first method involves treating both variables in continuous form. Examples include rubber sheet and soap film analogies (reference 13).

A second approach is the use of finite difference techniques applied to both independent variables. This is the same technique that is used when the equation is being solved on a digital computer. The region of interest is divided into a mesh and a system of equations is written for each node. The total number of nodes is  $N^2$  if the range of each independent variable is sub-divided into  $N$  equal intervals. The application of this scheme is not feasible on the analog computer since equipment requirements become excessive.

The third approach to the solution of partial differential equations on the analog computer is discretizing one variable while the other is left in continuous form. This method is the most practical of the three described for utilization on the analog computer.

In the next two sections, the parallel and the serial method will be described. In each of the methods, finite difference techniques are used to express one of the



independent variables in discrete form.

### 5-1. PARALLEL METHOD

In the parallel method, the range of the  $x$  variable is divided into  $N$  equal intervals. A separate computer set-up is then programmed for each of the  $N-1$  interpolation points. The solution is obtained simultaneously for each of the interpolation points. The advantage of this method is that the boundary conditions imposed on a closed region may be taken into account automatically without requiring any iteration technique to find a remaining boundary condition on the solution. The unfavourable aspect of using this method is that if improved accuracy is desired, equipment requirements become excessive. The general scheme involved in the parallel solution of partial differential equations is given in Figure 5-1.

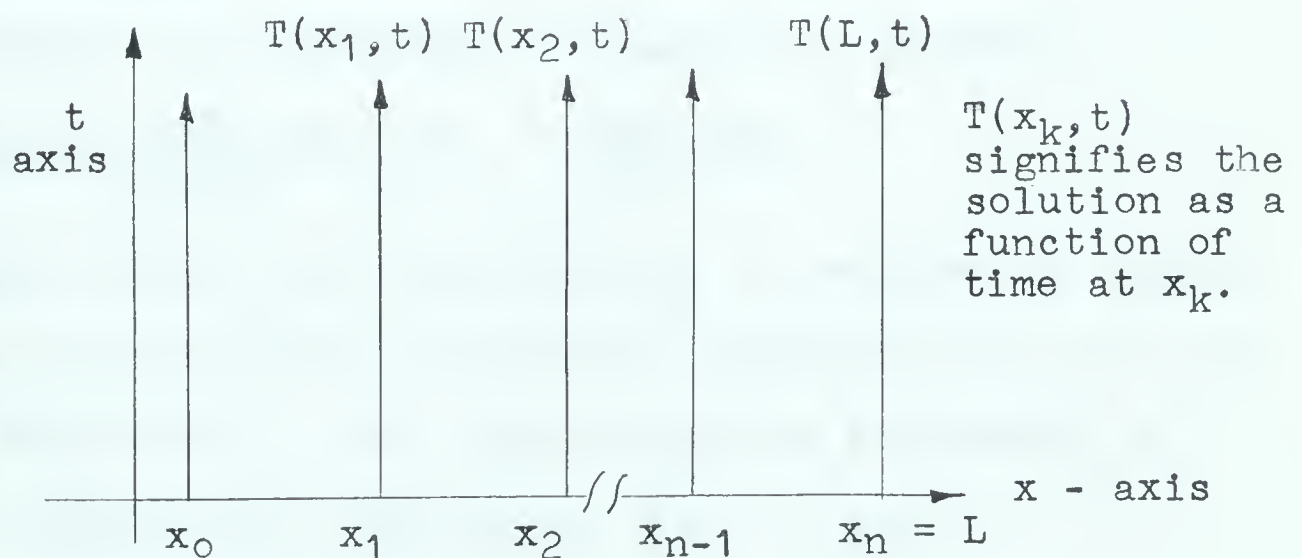


FIGURE 5-1. PARALLEL SOLUTION OF A PARTIAL DIFFERENTIAL EQUATION.



For an example, consider the application of the parallel method to the solution of a problem in heat conduction.

### Problem

A rod with a length  $L$ , with isolated lateral boundaries has the end  $x = 0$  maintained at  $T = T_1$  when  $t > 0$ , while the end at  $x = L$  is maintained at  $T = 0$ . The initial temperature distribution along the rod is prescribed as  $T(x,0) = f(x)$ . The equation governing the distribution of heat in the rod is given by;

$$\frac{\partial^2 T(x,t)}{\partial x^2} = K \frac{\partial T(x,t)}{\partial t} \quad (K = 1)$$

### Solution

Divide the length  $L$  into  $N$  equal intervals. The length of each interval is then  $L/N$ . We will designate  $x_k$  to mean  $x = kL/N$ . If we discrete the portion dependent upon  $x$ ;

$$\frac{T(x_{n+1},t) - 2T(x_n,t) + T(x_{n-1},t))}{(\Delta x)^2} = \frac{dT(x_n,t)}{dt}$$

If the equation is considered at station  $x_1$  we have;

$$\frac{T(x_2,t) - 2T(x_1,t) + T_1}{(\Delta x)^2} = \frac{dT(x_1,t)}{dt}$$

The computer set-up for this equation at station  $x_1$  (which also takes into account the boundary condition at  $x = 0$ ) is given in Figure 5-2. The computer may be programmed in a similar fashion for stations  $x_2, x_3, \dots, x_{n-1}$ .



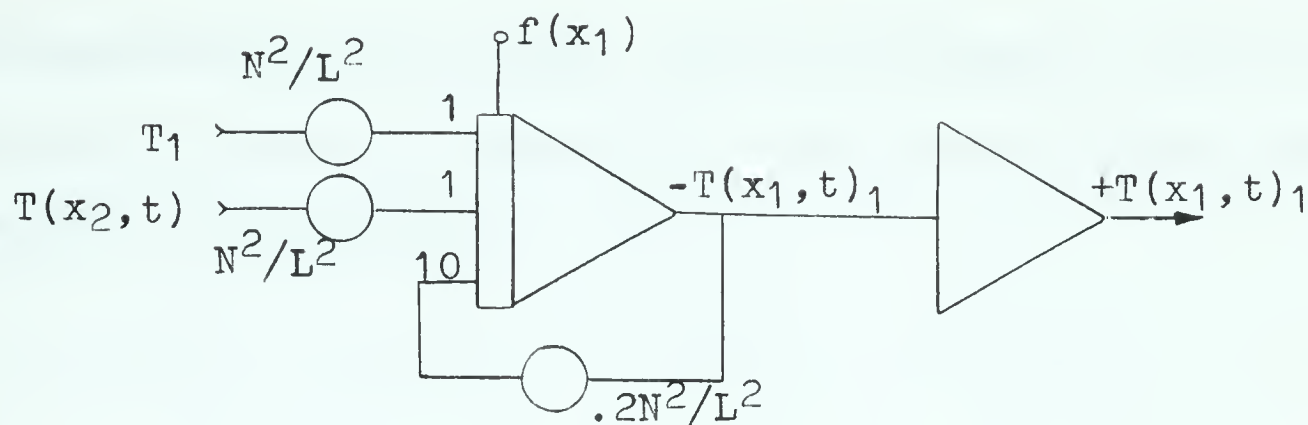


FIGURE 5-2. COMPUTER SET-UP FOR STATION  $x_1$ .

## 5.2 THE SERIAL METHOD

In the serial method of computation, the portion of the partial differential equation dependent on time is discretized. Then in the calculation for the solution corresponding to  $t + \Delta t$ , the solution for time  $t$  must be available. That is, a means of function storage must be available. The difficulty in employing this method is that if the partial differential equation is being solved on a closed region, an iteration scheme must be devised to find the necessary initial condition at  $x = 0$ . This does not pose a very serious problem if storage is available on the analog computer. The marked advantage of employing this method is that the equipment is multiplexed. The same computer program used for solving the discretized equations at time  $t$  is also used for the next time instant,  $t + \Delta t$ . Equipment requirements are less severe than in the parallel method. By decreasing the size of  $\Delta t$  more solutions will be obtained in a given region of interest, however there will be an increase in any inherent instabilities.



The scheme used in the serial solution of partial differential equations is given in Figure 5-3. The arrows indicate the direction of integration.

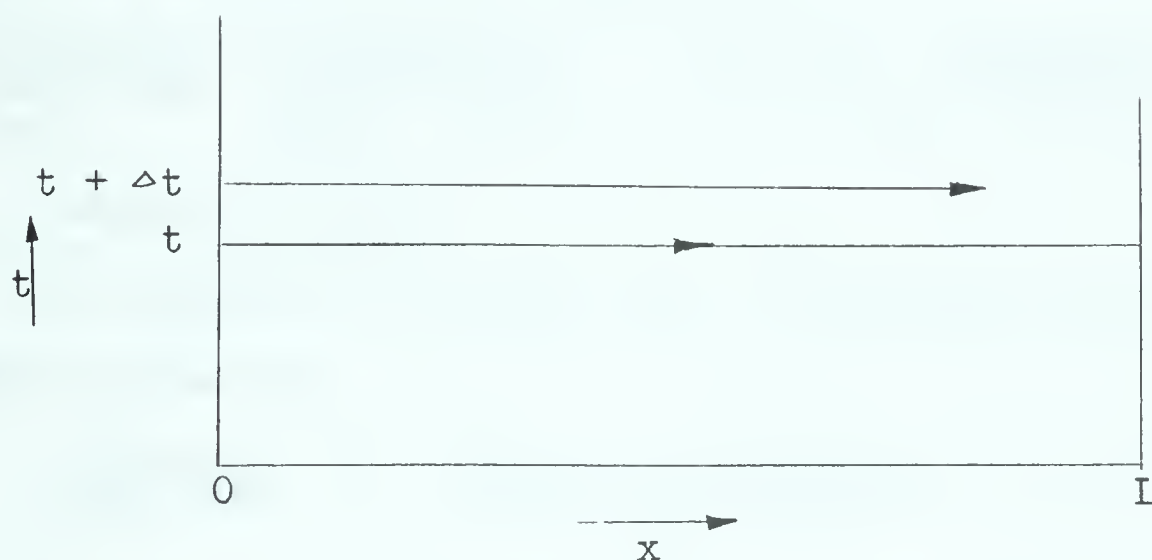


FIGURE 5-3. INTEGRATION PROCEDURE IN THE SERIAL METHOD

### 5-3. SOLUTION OF A HEAT PROBLEM USING THE SERIAL METHOD

The serial method will now be applied to a problem in heat conduction using iterative techniques on the analog computer. The problem that will be considered is given below.

A rod with length  $L$ , with isolated lateral boundaries has ends  $x = 0$  and  $x = L$  maintained at  $T = 0$  when  $t > 0$ . The initial temperature distribution along the rod is prescribed as  $T(x, 0) = 50 \sin \frac{\pi x}{L}$ . The equation governing the temperature distribution in the rod is given by;

$$\frac{\partial T(x, t)}{\partial t} = K \frac{\partial^2 T(x, t)}{\partial x^2}$$

$$K = L = 1 \quad (\text{numerically})$$



### 5.3.1 BLOCK DIAGRAM FOR THE ANALOG COMPUTER

If we now discrete the portion of the partial differential equation dependent upon time we have;

$$\frac{1}{2} \left[ \frac{d^2 T(x, t_n)}{dx^2} + \frac{d^2 T(x, t_{n-1})}{dx^2} \right] = \frac{T(x, t_n) - T(x, t_{n-1})}{\Delta t} \dots (5-1)$$

where  $t_n = n\Delta t$

If the interval  $\Delta t$  is made small, expression (5-1) may be approximated by:

$$\frac{d^2 T(x, t_n)}{dx^2} = \frac{T(x, t_n) - T(x, t_{n-1})}{\Delta t} \dots (5-2)$$

The block diagram for the solution of equation (5-2) using iterative techniques on the analog computer is given in Figure 5-4. The required function storage is performed as described in Chapter 4. The computer program for the X-Memory, the O-Memory, their respective controls, and for Newton's Iterative Function Generator is given in Chapter 4.

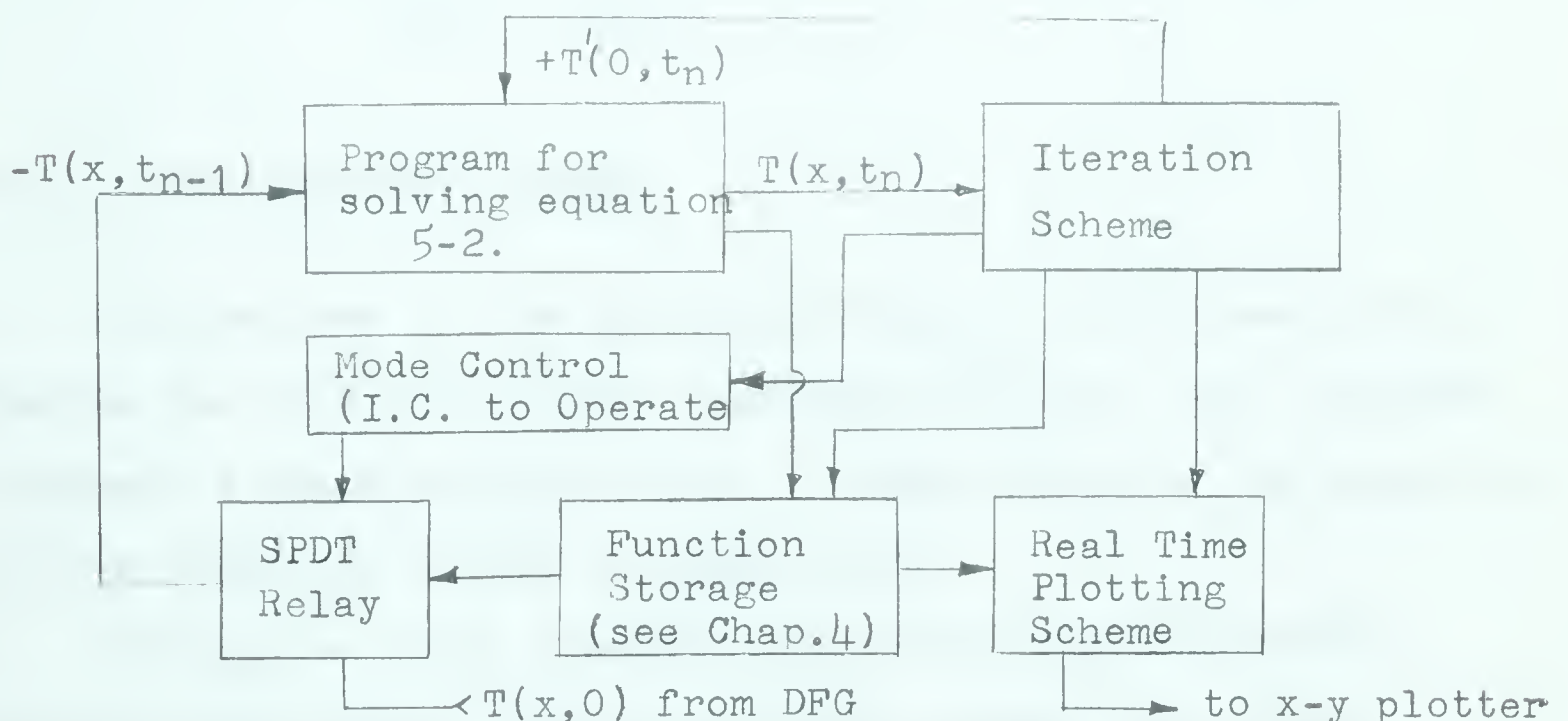


FIGURE 5-4. BLOCK DIAGRAM FOR SOLUTION OF HEAT PROBLEM.



### 5.3.2 COMPUTER PROGRAM FOR SOLUTION OF EQUATION (5-2)

The computer program for the solution of equation (5-2) is given in Figure 5-5.

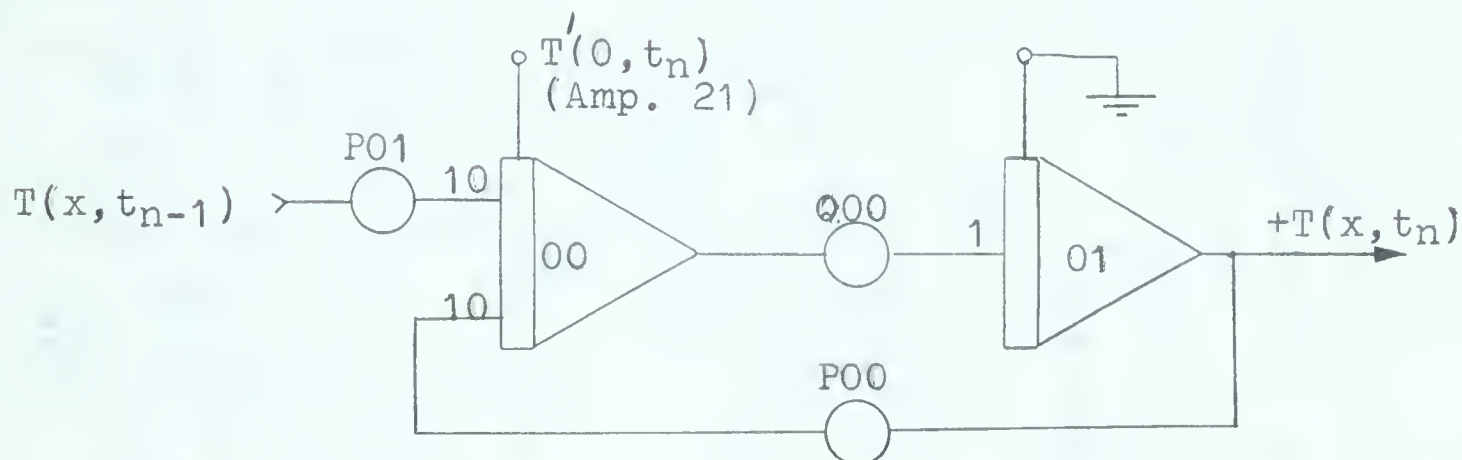


FIGURE 5-5. COMPUTER DIAGRAM FOR EQUATION (5-2).

Results were obtained for  $\Delta t = .0625$ . The potentiometer setting used for this particular value  $\Delta t$  are as follows:

Q00 - .1000

P00 - .1600

P01 - .1600

### 5.3.3 THE ITERATION SCHEME

The purpose of the iteration scheme is to automatically derive the required initial condition,  $T'(0, t_n)$ . The computer program is given in Figure 5-6. A description of the operation of the iteration scheme is given below.

During the first operate cycle where the temperature distribution for the time  $t_n$  is being sought, an initial guess for  $T'(0, t_n)$  is provided by amplifier 21. The output



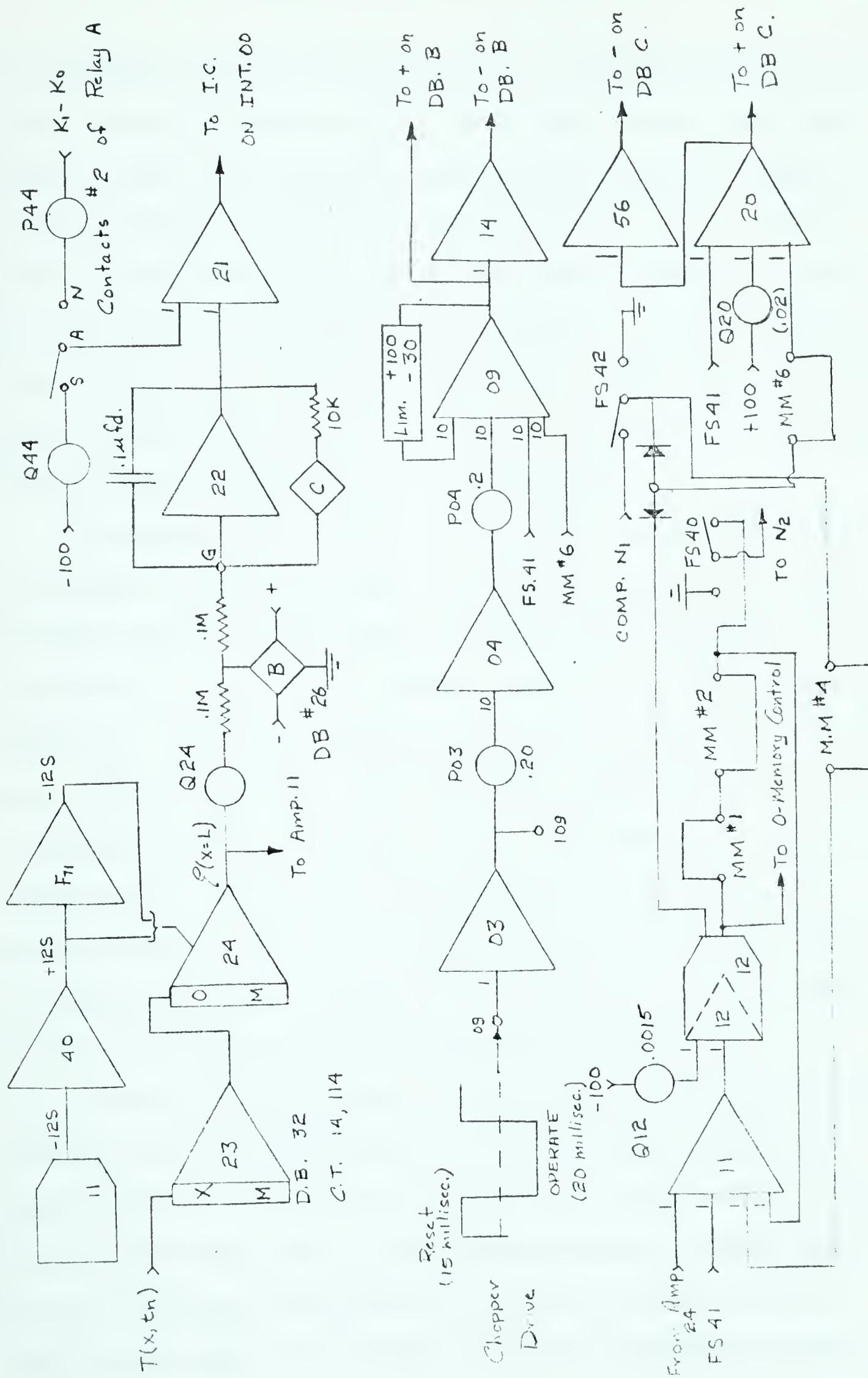


FIGURE 5-6. COMPUTER ITERATION SCHEME.



of integrator 22 can be assumed to be zero because of the reset operation which will be described later. The input to Amplifier 21 comes from the armature of contacts number 2 of relay A, which is driven by comparator  $N_1$ . As a result, the input to amplifier 21 is an initial guess adjusted by  $Q_{44}$  in the case where  $T'(0, t)$  is being sought, or  $(K_1 - K_0)|_{t_{n-1}}$  which serves as a good approximation for  $T'(0, t_n)$ . Potentiometer  $P_{44}$  adjusts the initial guess so that it is slightly less than the required initial condition.

Throughout the operate cycle, x-memory 23 will track the output voltage  $T(x, t_n)$ . At  $x = L$ , the mode of x-memory 23 will change and the value of  $T(L, t)$  for the first trial solution will be stored. This value will be designated as  $T(L, t_n)_1$ . Since the required value of  $T(L, t_n) = 0$ ,  $T(L, t_n)_1$  will serve as an error for the first iteration and will be designated as  $\xi_1$ . O-memory 24 will now track  $-\xi_1$  held by x-memory 23. Read in can take place for the first ten milliseconds of the reset cycle, because the mode control for x-memory 23 and o-memory 24 is identical to the X-Memory control for the function storage (see chapter 4).

During the reset cycle, diode bridge #26 will be non-conducting, hence integrator 22 will integrate at a rate proportional to the error  $\xi_1$ , held by o-memory 23. At the end of the reset cycle, diode bridge #26 will again conduct heavily, causing integrator 22 to hold its last value. Amplifier 21 will now provide a better approximation for



$T'(0, t_n)$  for the next operate cycle. In this iterative fashion the correct initial condition for  $T(x, t_n)$  will be obtained. A photograph of the performance of the iterating integrator 22 is given in Figure 5-12b.

Amplifier 09 and 14 provide mode control for integrator 22. A large voltage is used to place the diode bridge into heavy conduction so as to minimize the error due to drift when the integrator is in the hold state (see Appendix F). During the operate mode of the integrator (when bridge B is non-conducting), a smaller voltage is used to place the diode bridge into the non-conducting state in order to reduce errors due to unbalance in the diode bridge.

After  $k$  iterations, when the desired solution has been obtained, that is,  $T(L, t_n)_k = 0$ , comparator 12 will switch state. This will cause the o-memory in the function storage to read in the values of  $T(x, t_n)$ . The comparator will remain in this state for 8 milliseconds at which time MM #1 will trigger MM #2. This will place an inhibit pulse on comparator 12 insuring that this comparator switches back to the normal state and remains in this state for the remainder of the reset cycle, as well as for the next operate cycle, during which an iteration is started for obtaining the solution  $T(x, t_{n+1})$ .

At the instant comparator 12 switches state, MM #6 is triggered, producing a pulse of 14 millisecond duration. Amplifiers 20 and 56 place diode bridge C into the conducting



state for the duration of the pulse. This will cause integrator 22 to reset to zero since the output voltage from o-memory 24 is zero when the desired solution has been obtained.

The output voltage from amplifier 21 is now equal to  $(K_1 - K_0)|_{t_n}$  and the output from Newton's Iterative Function Generator is  $T(x, t_n)$ . The computer will now iterate for the solution  $T(x, t_{n+1})$  in a manner analogous to that described for  $T(x, t_n)$ .

#### 5.3.4 MODE CONTROL (I.C. TO OPERATE)

Mode control is required in the solution of the heat problem to change the input (corresponding to  $T(x, t_{n-1})$  in Figure 5-5) to the first integrator of the system solving Equation (5-2). This input is changed from the diode function generator output which generates the temperature distribution at  $t = 0$ , to the output from Newton's Iterative Function Generator. That is, when the solution for  $T(x, \Delta t)$  has been obtained, the mode of the system must be changed so that the remaining solutions  $T(x, k\Delta t)$  may be obtained. The computer program for this mode control is given in Figure 5-7, followed by a description of its operation.



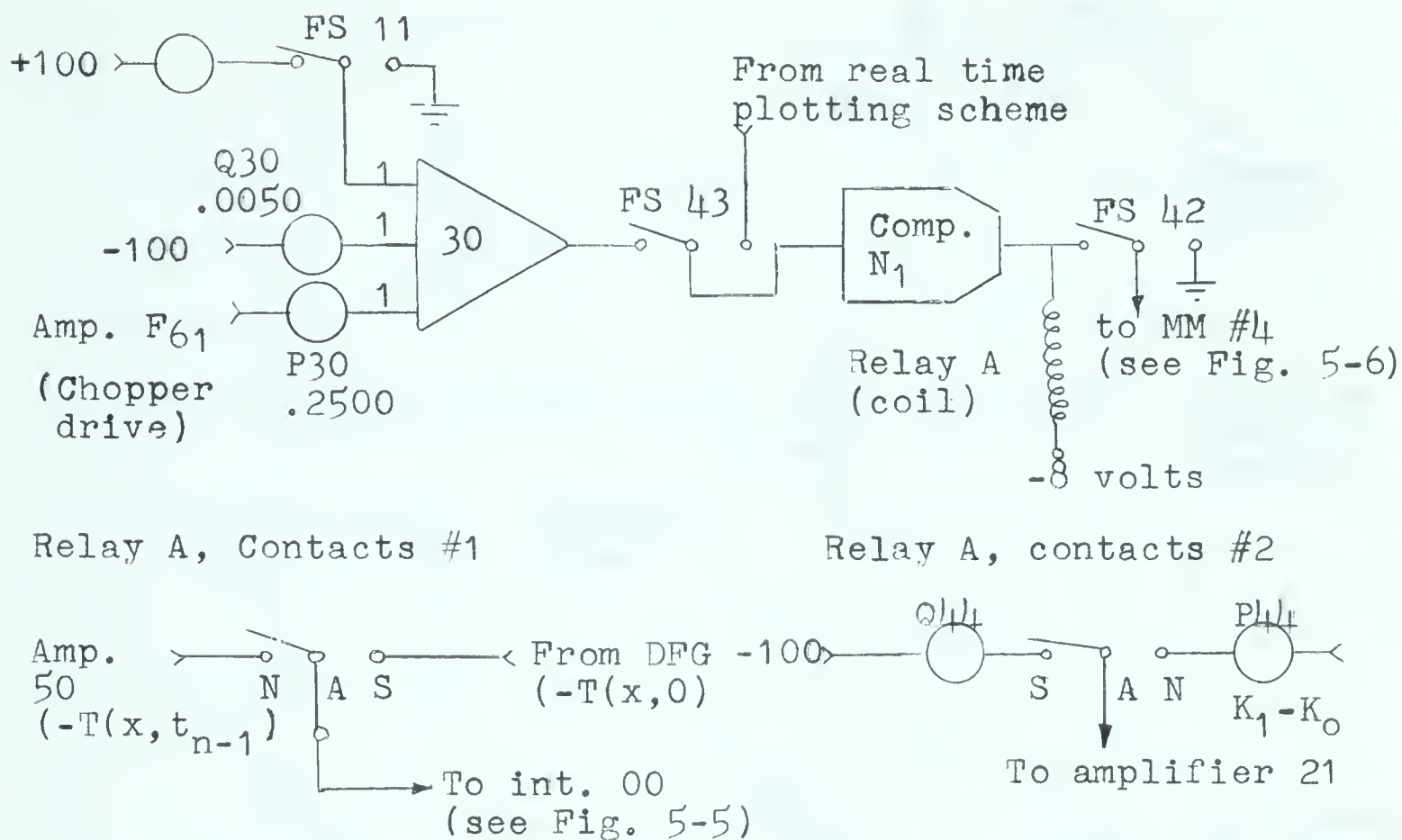


FIGURE 5-7. COMPUTER PROGRAM FOR MODE CONTROL.

The operation of the mode control can be best explained by considering the input voltage to the comparator  $N_1$ . This is depicted graphically in Figure 5-8.

If function switch 11 is closed, the input to comparator  $N_1$  will always be negative. This comparator will remain in what will be referred to as the switched state. The input to integrator 00 (see Figure 5-5) will be  $-T(x,0)$ , the initial temperature distribution in the rod. When the iterations for the solution of  $T(x,\Delta t)$  are completed, the iteration scheme will reset and the resulting function  $T(x,\Delta t)$  will be read into the 0-Memory of Newton's Iterative Function Generator. However, since the input to integrator 00 will



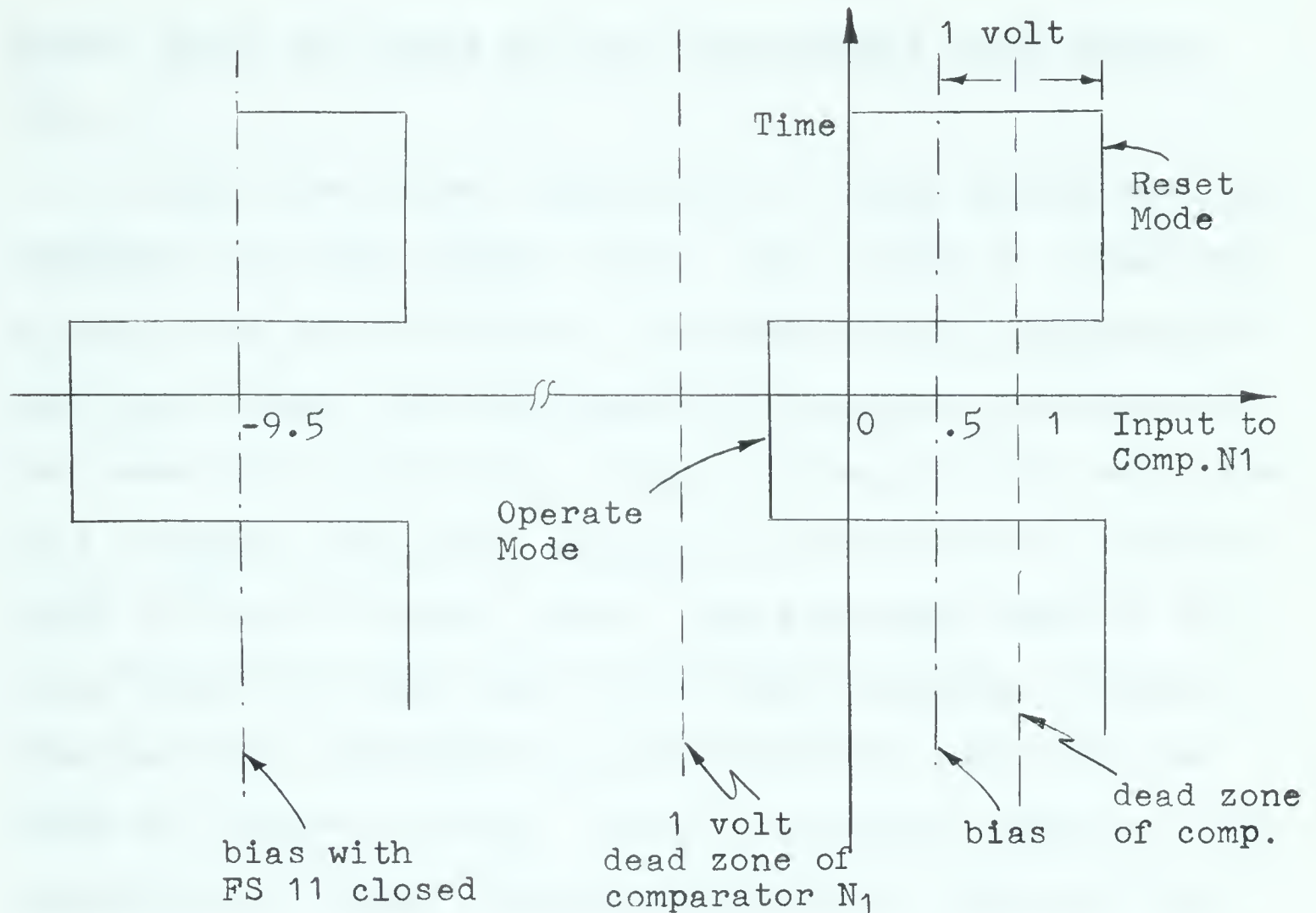


FIGURE 5-8. INPUT VOLTAGE TO COMPARATOR  $N_1$ .

still remain  $T(x,0)$ , this same solution will be continually solved until the state of relay A is changed.

To obtain the solution  $T(x,2\Delta t)$ , the state of comparator  $N_1$  must be changed so that when  $T(x,\Delta t)$  is obtained, the state of relay A will be such that  $-T(x,\Delta t)$  will be feed into integrator 00 for the next operate cycle. However, this transition must occur only during the reset cycle if the data stored in the 0-Memory is not to be destroyed. The logic required for accomplishing this operation is performed by amplifier 30 and comparator  $N_1$ . Function switch 11 may be opened while the computer is either in the operate or the reset



mode. These two cases are now investigated using Figure 5-8.

In the first case, consider FS 11 being opened when the computer is in the operate mode. The voltage to comparator  $N_1$  will rise to  $-0.5$  volts. The state of the comparator  $N_1$  will not change. At the instant the computer switches into the reset mode, the input voltage to comparator  $N_1$  will rise to  $1.5$  volts. This will cause  $N_1$  to switch state, driving relay A into its normal state. The switching speed of the relay used is in the order of five milliseconds, therefore there will be sufficient time in the reset cycle for the relay to change its state. When the computer returns to the operate mode (chopper drive voltage becomes  $-4$  volts) the input voltage to comparator  $N_1$  becomes  $-0.5$  volts. However, because of the one volt dead-zone, comparator  $N_1$  will remain in its normal state.

In the second case, FS 11 is opened when the computer is in the reset mode. No complications arise in this case. Comparator  $N_1$  will switch to the normal state and remain in this state as described in the previous paragraph.

When FS 11 is opened, no information is available as to what state the iteration scheme is in. However, within  $0.1$  seconds the first solution  $T(x, \Delta t)$  is obtained and is stored in the O-Memory. At the instant comparator  $N_1$  switches state, MM #6 is triggered, resetting integrator 22. An inhibit pulse is also placed on comparator 12 by MM #4 insuring



that <sup>at</sup> least one more iteration is performed before this comparator will switch state. The input to integrator 00 is now equal to  $T(x, \Delta t)$ , which is obtained from the Newton's Iterative Function Generator. The computer will begin iterations for  $T(x, 2\Delta t)$ . The remaining solutions  $T(x, k\Delta t)$  are obtained in a manner described in Section 5.3.3.

#### 5.3.5 REAL TIME PLOTTING SCHEME

This scheme was devised in order to obtain results  $T(x, k\Delta t)$  on an x-y plotter. The essence of the scheme is as follows:

When a solution for a new increment of time is obtained, this solution is first read into the O-Memory of Newton's Iterative Function Generator. An inhibit pulse is then placed on Integrator 22 so that no convergence to the solution for the next increment of time is obtained. A sampled data plotting scheme similar to that described in reference 11 is then used to read out the function stored in the O-Memory. Five seconds are allowed for the x-y plotter to record the solution and two seconds for the reset of the plotter. The inhibit pulse on integrator 22 is then removed while the computer is in the reset mode and iterations for the solution for the next interval of time will begin. The entire solution  $T(x, k\Delta t)$  may be obtained on an x-y plotter by a repetition of the above process. The computer program for the real time plotting scheme is given in Figure 5-9. The prefix E will



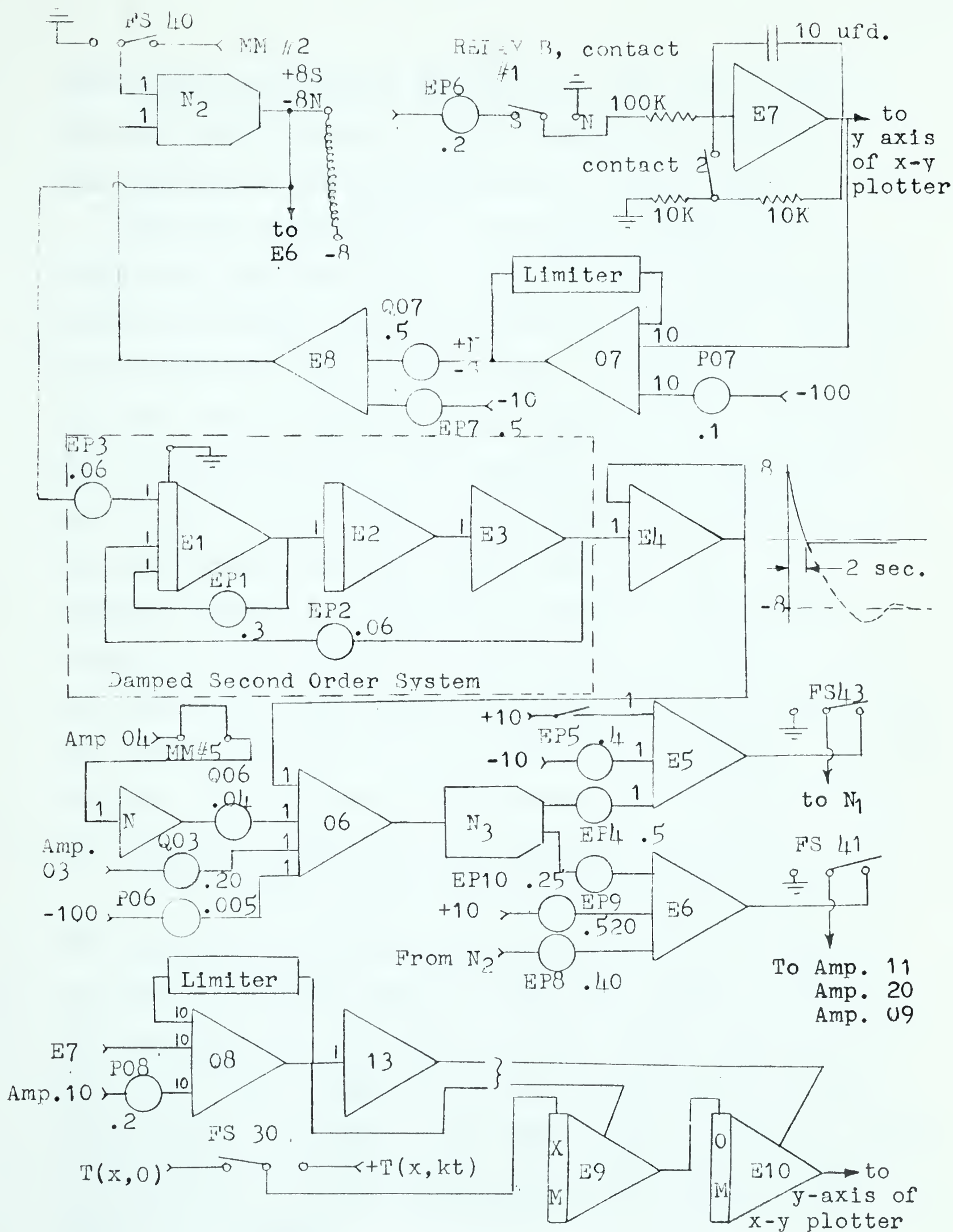


FIGURE 5-9. REAL TIME PLOTTING SCHEME.



imply that the equipment designated in this way is part of the Pace TR-10 computer. A more detailed description of the operation of the plotting scheme is given below.

When the iteration for a particular solution is completed, comparator 12 will switch state, triggering MM #1. Eight milliseconds later MM #2 will be triggered. This will cause comparator  $N_2$  to change state, driving relay B to the switched state. Integrator E7 will now produce a ramp which will reach ten volts in five seconds. This output serves as a drive for the x-axis of the plotter and as an input for the sampled-data plotting scheme comparator 08. As the computer produces the solution in repetitive operation, x-memory E9 will sample the function  $T(x, k\Delta t)$  at successive intervals of  $x$ . The o-memory will hold or store these values and serve as the input to the y-axis of the plotter. In five seconds, 1,430 samples of the function will be taken. Thus, the output of o-memory E10 will approach a continuous curve.

At the instant comparator  $N_2$  switches state, amplifier E6 will place an inhibit pulse on integrator 22 causing it to reset to zero and remain in this state until the plot on the x-y plotter is completed. At the instant integrator E7 reaches 10 volts, comparator 07 will change state causing comparator  $N_2$  to return to the normal state. This will cause integrator E7 to reset to zero volts with a time constant equal to .1 seconds.

The input voltage from amplifier E4 will cause comparator  $N_3$  to provide an inhibit pulse on Integrator 22 for an additional two seconds. The operation of comparator



$N_3$  will be explained by reference to Figure 5-10, which gives the input voltage to this comparator.

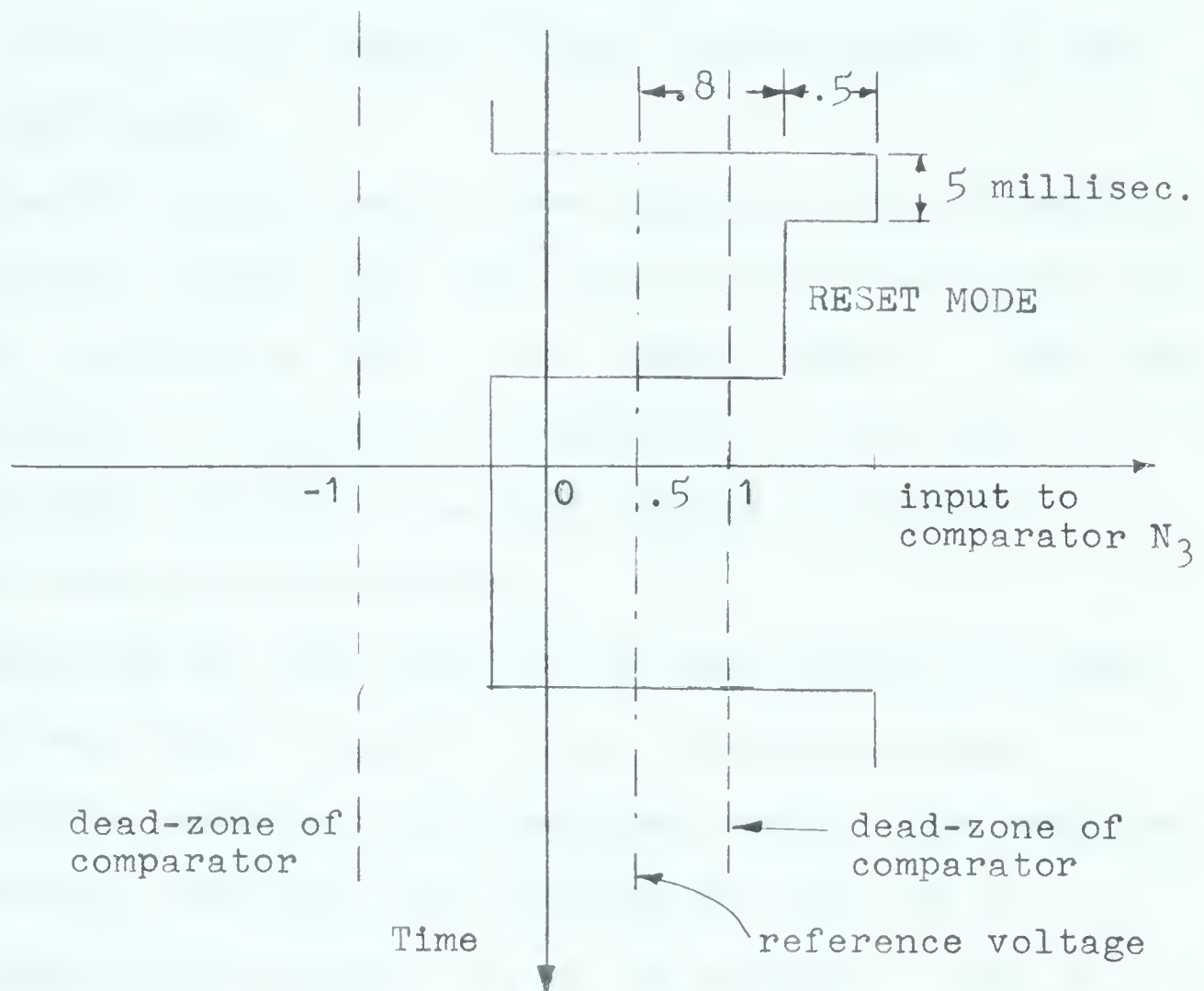


FIGURE 5-10. INPUT VOLTAGE TO COMPARATOR  $N_3$ .

The reference voltage to comparator  $N_3$  at the instant comparator  $N_2$  switches back to the normal state is -7.5 volts. This reference voltage will decay to +.5 volts in 2 seconds. This response is obtained from the program of a damped second order system. Superimposed on the chopper drive is a pulse of 5 millisecond width which is derived from MM #5. The purpose of this pulse is to insure that when the inhibit pulse is removed, it is in the first 5 milliseconds of the



reset cycle. Otherwise if switching occurs in the latter part of the reset cycle, a risk of destroying the stored function exists. When comparator  $N_3$  switches back to its normal state it will remain in this state because of the 1 volt dead zone.

When the inhibit pulse from comparator  $N_3$  is removed, the iteration scheme will proceed to find the solution for the next interval of time in the normal fashion. When the next solution is obtained, the sequence of operations for plotting this solution on an x-y plotter is identical to that for the previous solution.

Amplifier E5 and comparator  $N_1$  now perform the mode control described in Section 5.3.4. With  $S_1$  closed, comparator  $N_1$  remains in the switched state. The computer will continue obtaining the solution  $T(x, \Delta t)$ . By use of FS 30 both  $T(x, 0)$  and  $T(x, \Delta t)$  may be plotted. With  $S_1$  open, no change in the state of comparator  $N_1$  will occur until the inhibit pulse from comparator  $N_3$  is removed. A better explanation for the requirement that  $N_3$  must switch in the first 5 milliseconds of the reset cycle may be given. The switching time for relay A driven by comparator  $N_1$  is in the order of 5 milliseconds. Therefore a safety factor of 5 milliseconds exists for relay A to completely switch in the reset cycle. If switching is not completed in the reset cycle the solution  $T(x, \Delta t)$  could be destroyed, rendering the remainder of the solutions meaningless.



### 5.3.6 INITIAL CONDITION $T(x,0)$

The initial temperature distribution in the rod is generated using a diode function generator. The curve is approximated using 16 segments. The computer program is given in Figure 5-11.

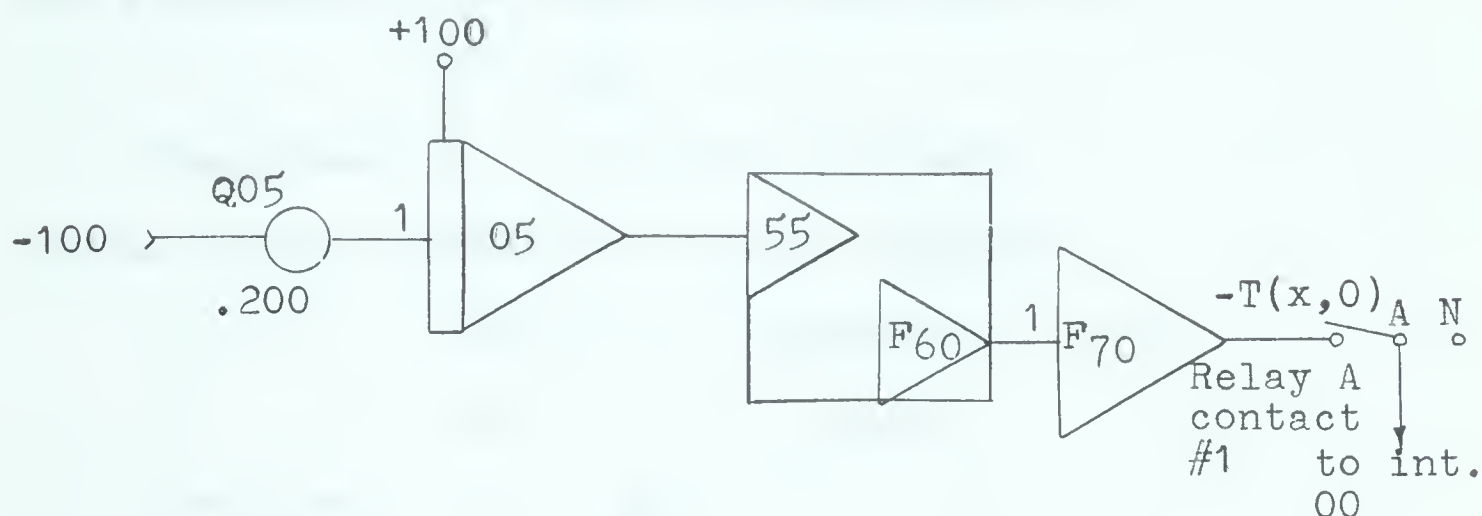


FIGURE 5-11. GENERATION OF INITIAL CONDITION  $T(x,0)$ .

### 5.4 THEORETICAL RESULTS

The separation of variables method will be applied to the following partial differential equation.

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\partial T(x,t)}{\partial t} \quad 0 \leq x \leq L = 1$$

$$\text{with } T(x,0) = 50 \sin x/L$$

$$T(0,t) = 0$$

$$T(L,t) = 0$$

Assume a solution of the form;  $T(x,t) = T(t)X(x)$ . Substituting this in the partial differential equation we have;

$$X''T = XT'$$



$$\frac{T'}{T} = \frac{X''}{X} = -\lambda^2$$

We now have two ordinary differential equations. The first one is;

$$T + \lambda^2 T = 0$$

$$\text{therefore } T(t) = c_1 e^{-\lambda^2 t}$$

The solution for the space dependent part is

$$X'' + \lambda^2 X = 0$$

$$\text{therefore } X = c_2 \sin \lambda x + c_3 \cos \lambda x$$

If the initial conditions are now applied:

$$X(0) = 0 \quad \text{therefore } c_3 = 0$$

$$X(L) = 0 = c_2 \sin \lambda L$$

$$\text{therefore } \lambda L = \frac{n\pi}{L}$$

The complete solution is;

$$T(x,t) = c_n \sin \frac{n\pi x}{L} e^{-\frac{n^2 \pi^2 t}{L^2}}$$

$$T(x,0) = c_n \sin \frac{n\pi x}{L}$$

But this is just the Fourier Series for  $T(x,0)$  where the coefficients  $c_n$  are given by

$$c_n = \frac{2}{L} \int_0^L T(x,0) \sin \frac{n\pi x}{L} dx$$

Substituting  $L = 1$ , and  $T(x,0) = 50 \sin \pi x$ ,

$$c_n = 2 \int_0^1 50 \sin \pi x \sin n\pi x dx$$

The integral is non-zero iff  $n = 1$ , at which time

$$c_1 = 100 \int_0^1 \sin^2 \pi x dx = 100 \int_0^1 \frac{(1 - \cos 2\pi x)}{2} dx = 50$$

The theoretical solution is,



$$T(x,t) = 50 e^{-\pi^2 t} \sin \pi x \dots\dots\dots (5-3)$$

## 5.5 COMPARISON OF THEORETICAL AND COMPUTER RESULTS

Photographs of results obtained on the analog computer are given in Figure 5-12c. The smallest value of  $\Delta t$  that could be chosen was .0625. The reason for this restriction will be given after the theoretical solution is obtained for the difference equation. Consider the following discretized equation which gives the temperature distribution at time  $t$ .

$$\frac{d^2 T(x,t)}{dx^2} = \frac{T(x,t) - T(x,t-\Delta t)}{\Delta t} \dots\dots\dots (5-4)$$

$$\text{Let } 1/\Delta t = a$$

The theoretical solution obtained for this problem showed that the temperature distribution for any time  $t$  was of the form,  $B \sin \pi x$ . Therefore we assume that,

$$T(x,t-\Delta t) = B \sin \pi x$$

$$T'(0,t) = A$$

Substituting the above quantities into equation (5-4) we have,

$$\frac{d^2 T(x,t)}{dx^2} = a T(x,t) = -a B \sin \pi x \dots\dots\dots (5-5)$$

If we now take the Laplace transform of equation (5-5);

$$s^2 T(s) - sT(0+) - T'(0+) - aT(s) = \frac{-aB\pi}{s^2 + \pi^2}$$

$$(s^2 - a)T(s) = A - \frac{aB\pi}{s^2 + \pi^2}$$

$$T(s) = \frac{A}{(s^2 - a)} - \frac{aB\pi}{(s^2 - a)(s^2 + \pi^2)}$$



Using Newton's partial fraction expansion method we obtain;

$$\begin{aligned}
 T(s) = & \frac{aB}{(\pi^2 + a)(s^2 + \pi^2)} - \frac{aB\pi}{2\sqrt{a}(a + \pi^2)} \frac{1}{(s - \sqrt{a})} \\
 & + \frac{aB\pi}{2\sqrt{a}(a + \pi^2)} \frac{1}{(s + \sqrt{a})} + \frac{A}{2\sqrt{a}} \frac{1}{(s - \sqrt{a})} \\
 & - \frac{A}{2\sqrt{a}} \frac{1}{(s + \sqrt{a})} \dots\dots\dots(5-6)
 \end{aligned}$$

The corresponding time solution for equation (5-6) is;

$$\begin{aligned}
 T(x,t) = & c_1 B \sin \pi x - c_2 B e^{+\sqrt{a} x} + c_2 B e^{-\sqrt{a} x} \\
 & + c_3 A e^{+\sqrt{a} x} - c_3 A e^{-\sqrt{a} x} \dots\dots\dots(5-7)
 \end{aligned}$$

where

$$\begin{aligned}
 c_1 &= \frac{a}{(\pi^2 + a)} \\
 c_2 &= \frac{a\pi}{(2\sqrt{a})(a + \pi^2)} \\
 c_3 &= \frac{1}{2\sqrt{a}}
 \end{aligned}$$

In order that the boundary condtion at  $x = 1$  be satisfied, (that is,  $T(1,t) = 0$ );

$$\begin{aligned}
 c_2 B &= c_3 A \\
 A &= \frac{c_2 B}{c_3}
 \end{aligned}$$

Equation (5-7) will be used to calculate the theoretical solution to the discrete equation (5-4). The actual solution to the problem will be obtained using equation (5-3). Table 5-1 gives a comparison between the computer solution, the theoretical solution to the discrete equation and the actual solution for a time increment of .0625 seconds. In Table 5-1, M will signify the maximum value of the sinusoidal solution for each interval of time.



Time (sec.)	Computer Results	Theoretical Sol <sup>n</sup> (using eqn.5-7) M	Theoretical Sol <sup>n</sup> (using eqn. 5-3) M
0	50	50	50
.0625	31	30.9	26.9
.1250	19	19.1	14.5
.1875	12	11.8	7.8
.2500	7	7.3	4.2

TABLE 5-1. COMPUTER AND THEORETICAL RESULTS.

In order that the solution of the discretized equation (5-4) give a closer approximation to the actual solution, a smaller increment in time must be chosen. The computer solution of the discretized equation compares very favorably with the theoretical solution obtained using equation (5-7). Using the real time plotting scheme the error can be shown to be less than .5 percent. A comparison between the solution of equation (5-4) and the actual solution given by equation (5-3) is given in Table 5-2 for  $\Delta t = .01$  seconds.

From Table 5-2, the error using the discrete equation (5-4) is 2.4 percent as compared to 15 percent for approximately the same time instant using  $\Delta t = .0625$  (see Table 5-1). If the increment in time is made smaller, even better results may be obtained.

An explanation of the reason why the increment in time could not be made less than .06 seconds will now be given.



Time (seconds)	Theoretical Solution (using equation 5-7) M	Theoretical Solution (using equation 5-3) M
0	50	50
.01	45.5	47.1
.02	41.4	41.1
.03	37.7	37.2
.04	34.3	33.6
.05	31.2	30.5
.06	28.4	27.6

TABLE 5-2. THEORETICAL RESULTS FOR  $\Delta t = .01$  .

A careful examination of equation (5-7) will show that if the value of A is corrupted by noise of a magnitude  $\delta$ , then the solution will deviate from the desired solution by  $\delta e^{\sqrt{a} x}$ . At the final value of x, namely  $x = 1$ , the solution will diverge by  $\delta e^{\sqrt{a}}$ . For the case of  $\Delta t = .0625$ , this deviation is equal to 548. An examination of the initial condition on integrator 00 showed a maximum possible error of .2 volts. This offers sufficient explanation why the solutions tend to "flutter" towards the end of the operate cycle when the computer is in repetitive operation. By making  $\Delta t$  even smaller, this error is excentuated even further.

The source of this error was traced to the choppers stabilizing the amplifiers in the computer. Mode control for the integrators was then built up using field-effect



transistors switched from a General Radio pulser. This pulser was triggered externally from the 94 cycle/sec. voltage driving the stabilization choppers. Results obtained using this external mode control showed no perceptible "flutter".

It may be concluded that the application of iterative techniques to the solution of partial differential equations on the analog computer represents a powerful method. Examining Table 5-2 shows that the error between the theoretical and the computer solution to the discrete equation (5-4) is almost insignificant. With external mode control implemented with field-effect transistors, much smaller values for the increment in time ( $\Delta t$ ) may be used, giving results well within one percent error. Using transistorized operational amplifiers (Nexus FSL-12) with mode control performed by field-effect transistors, iterative computation may easily be performed at 10 kilocycles per second (reference 14). Since these operational amplifiers use no choppers for stabilization, the error previously described will no longer exist. This would then present a practical technique for the solution of partial differential equations using iterative techniques at ultra-high speed on the analog computer.



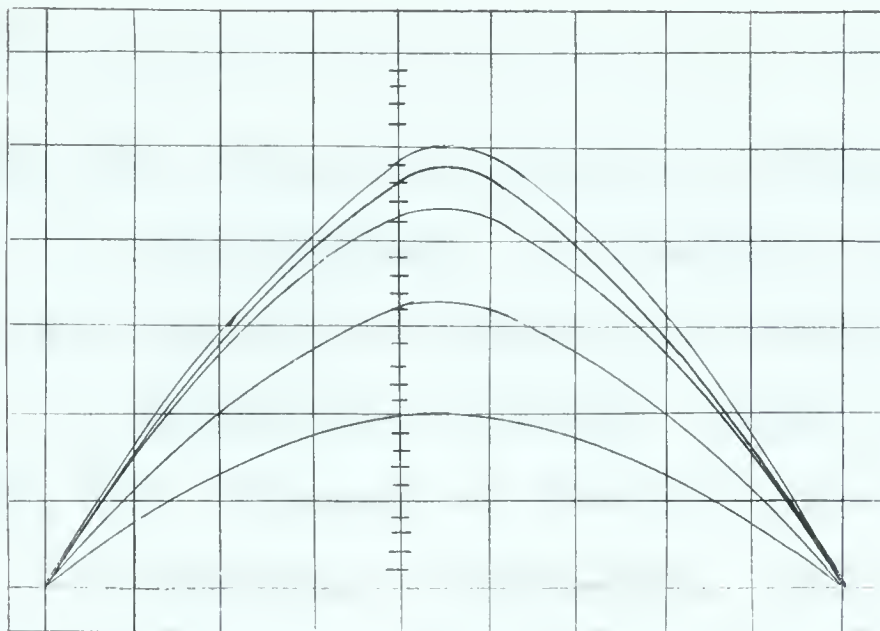


FIGURE 5-12a. ITERATIVE SOLUTION OF BOUNDARY VALUE IN O.D.E.

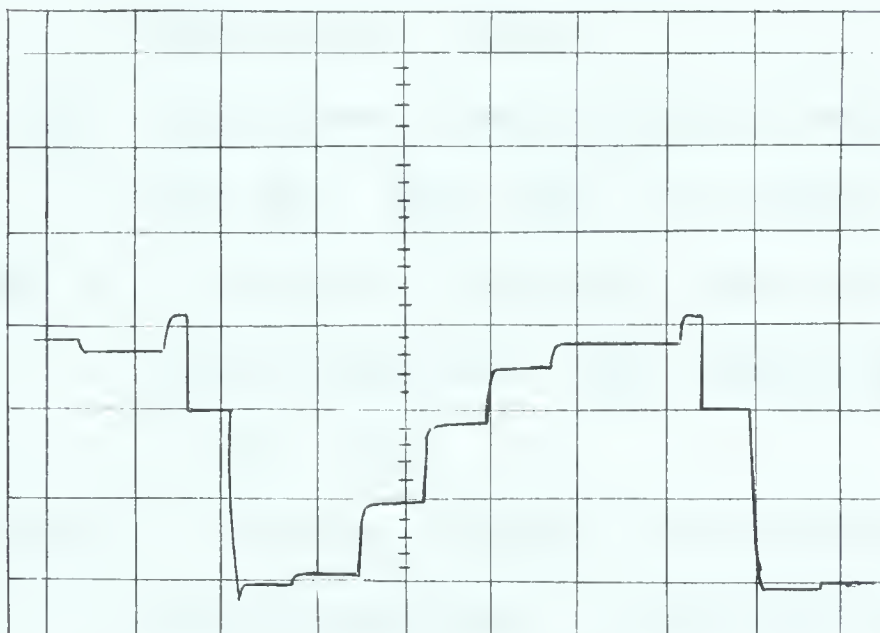


FIGURE 5-12b. PERFORMANCE OF ITERATING SCHEME.  
(INPUT TO COMPARATOR 12)

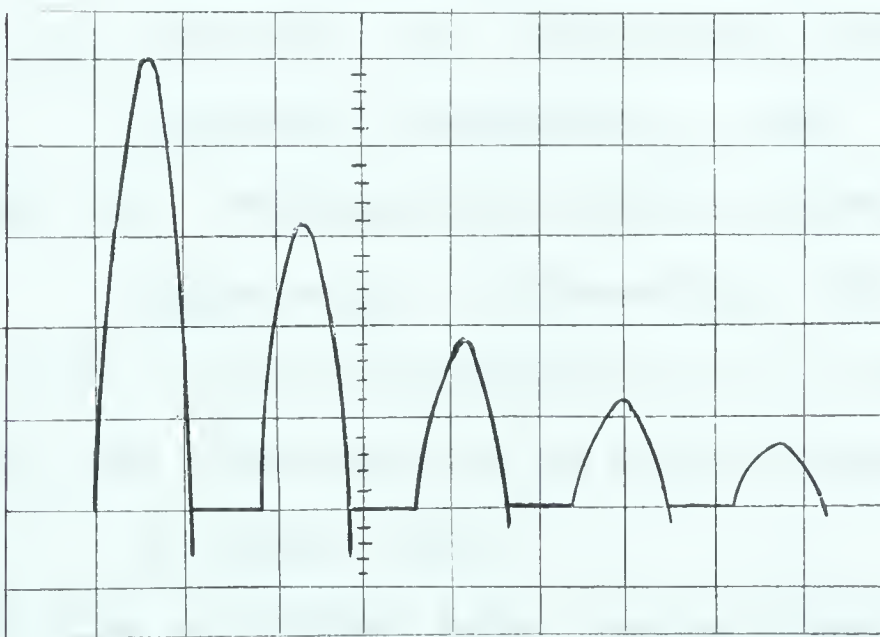


FIGURE 5-12c. SOLUTION OF HEAT PROBLEM ( $\Delta t = .0625$ ).



## BIBLIOGRAPHY

- (1) HAMMING, R.W: Numerical Methods for Scientists and Engineers. McGraw-Hill, 1962.
- (2) SAGAN, H: Boundary and Eigenvalue Problems in Mathematical Physics. Wiley, 1961.
- (3) SNEDDON, I.N: Elements of Partial Differential Equations. McGraw-Hill, 1957.
- (4) GREINER, R.A: Semiconductor Devices and Applications. McGraw-Hill, 1961.
- (5) JURY, S.H: High Speed Memory Analog Computer. Ind. Eng. Chem.-53, 173 (1961).
- (6) HANNAUER, G: Automatic Iterative Operation on the Analog Computer. EAI Computing Techniques 1.3.4a, 1963.
- (7) BERTHIAUM, P: Analog Computer Storage Techniques and Applications. Princeton Computation Center Report 170, 1961.
- (8) KORN, G.A. and KORN, T.M: Electronic Analog and Hybrid Computers, McGraw-Hill, 1964.
- (9) DETTMAN, J.W. Mathematical Methods in Physics and Engineering. McGraw-Hill, 1962.
- (10) KOZAK, W.S: Can. Electronics Eng. 38, 1958.
- (11) ASHLEY, J.R: Introduction to Analog Computation. J. Wiley, 1963.
- (12) MACKAY, D.M. & FISHER, M.E: Analog Computing at Ultra-High Speed. Chapman and Hall, 1962.



- (13) FIFER, S: Analog Computation, Vol.3. McGraw-Hill, 1961.
- (14) ASH, J.A: The use of field-effect transistors for  
implementing mode control for analog computers  
was demonstrated in the Electrical Engineering  
Department of the University of Alberta.



## APPENDIX A: ITERATIVE CONTROL UNIT

The Iterative Control Unit consists of the specialized equipment required for iterative computation on the analog computer. The unit includes twelve electronic comparators, three comparators with variable dead zone, thirty-six diode bridges, six monostable multivibrators, and one double pole double throw relay. This Iterative Control Unit also contains its own plus and minus twelve volt power supply (Dressen-Barnes Electronic Corporation Power Supply, model 20-12S). A front and side view of the Iterative Control Unit giving the equipment layout is given in Figure A-1.

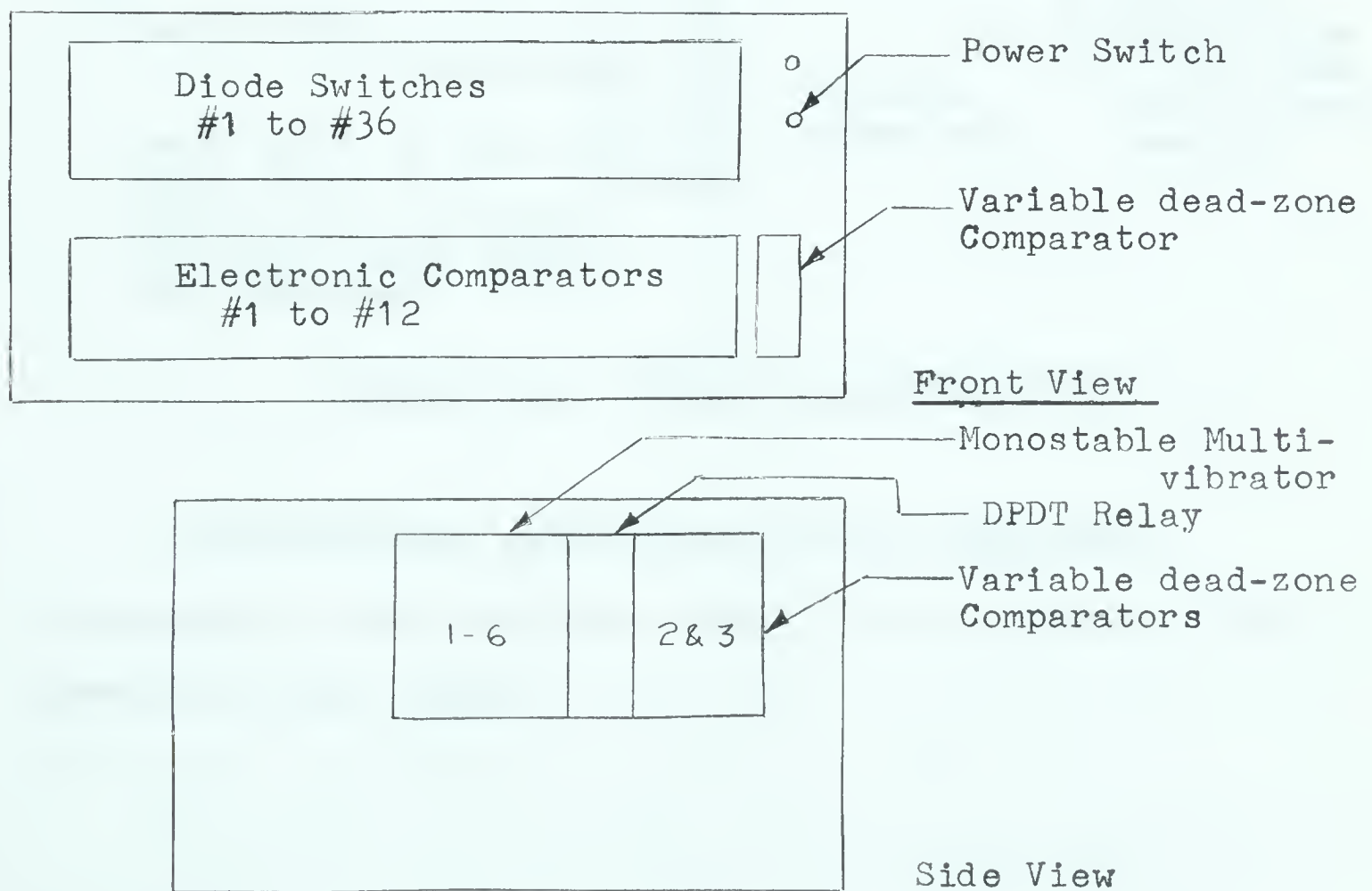


FIGURE A-1. ITERATIVE CONTROL UNIT.



The front panel of this unit has been drilled to accept the shielded leads used in patching on the Pace Computer. This facilitates direct patching between the computer patch panel and the Iterative Control Unit. Details of the method by which contact is made from the patch panel on the computer to the Iterative Control Unit is given in Figure A-2.

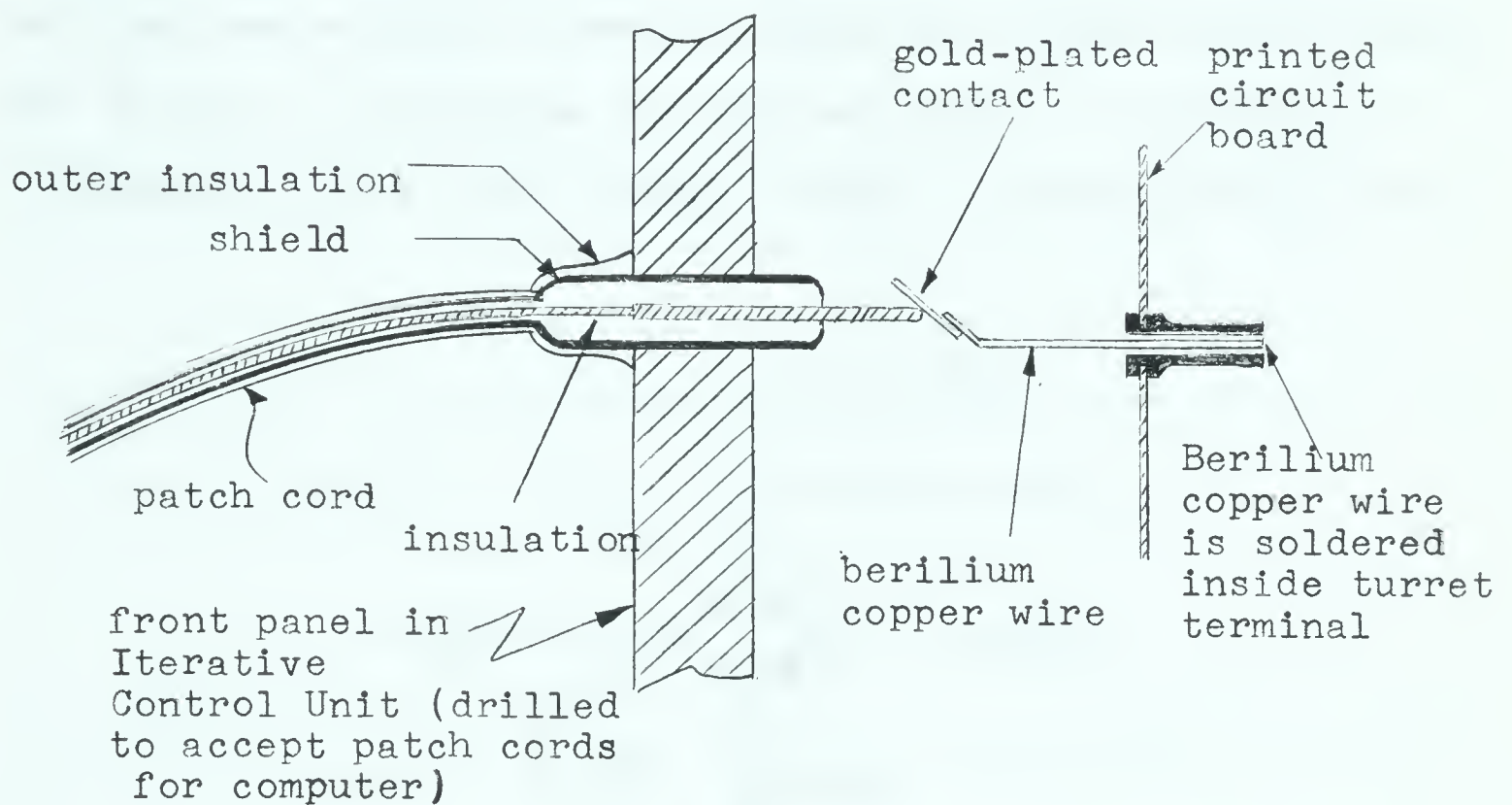


FIGURE A-2. CONTACT ARRANGEMENT.

A more detailed description of the individual components of the Iterative Control Unit is given in the Appendices that follow.



## APPENDIX B. DIODE BRIDGE SWITCH

The major component in implementing storage on an operational amplifier is the diode bridge switch. The need for an ideal electronic switch poses a serious problem in accurate iterative computation. In the diode bridges incorporated into the Iterative Control Unit, all resistors were matched within .1 percent while the forward drop across the diodes in conduction was matched within .5 millivolts. A schematic for a diode bridge switch is given in Figure B-1.

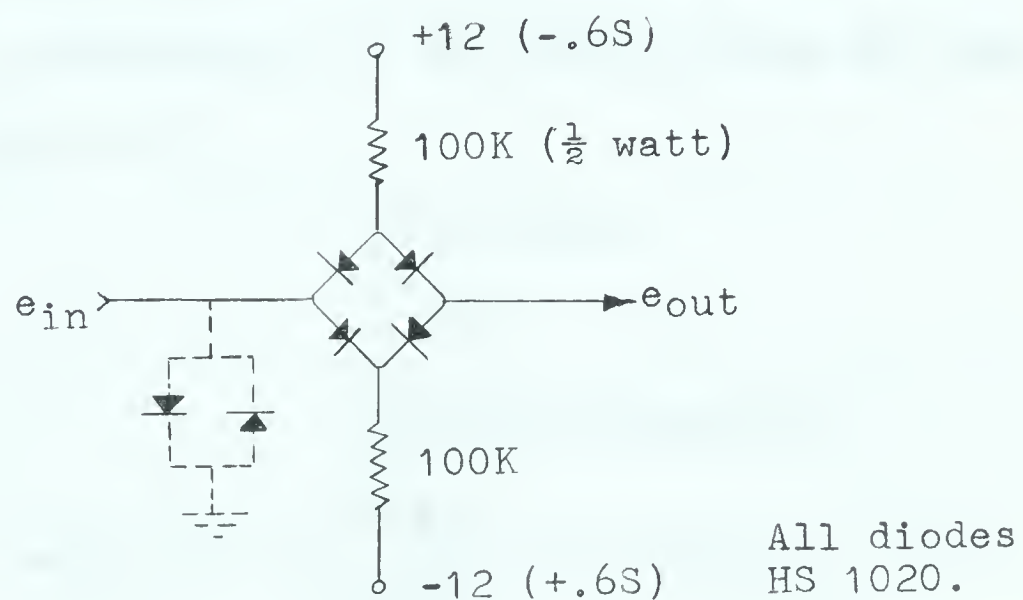


FIGURE B-1. DIODE BRIDGE SWITCH SCHEMATIC.

The scheme used for measuring the forward drop across the diodes in conduction is given in Figure B-2.

When used in a track-hold circuit, the error due to the unbalance in the diode bridge is less than 0.05 volts. However in the case where a diode bridge is used in the summing junction of an integrator (Switch A and B in Newton's



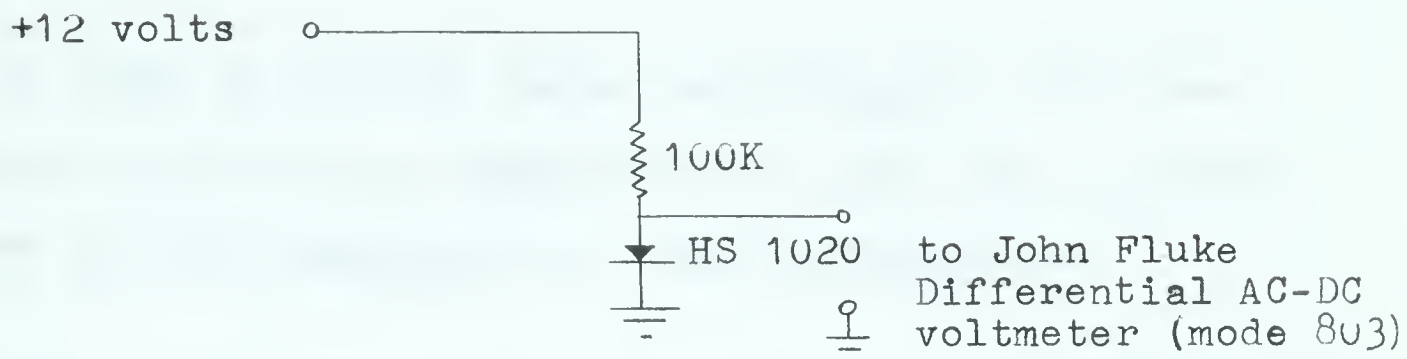


FIGURE B-2. MEASURING OF FORWARD RESISTANCE OF DIODE.

Iterative Function Generator) an unbalance of .05 volts integrated for ten seconds gives an error of .5 volts. To reduce the unbalance in these particular diode bridges even further, a modification to the diode bridge was made as indicated in Figure B-3.

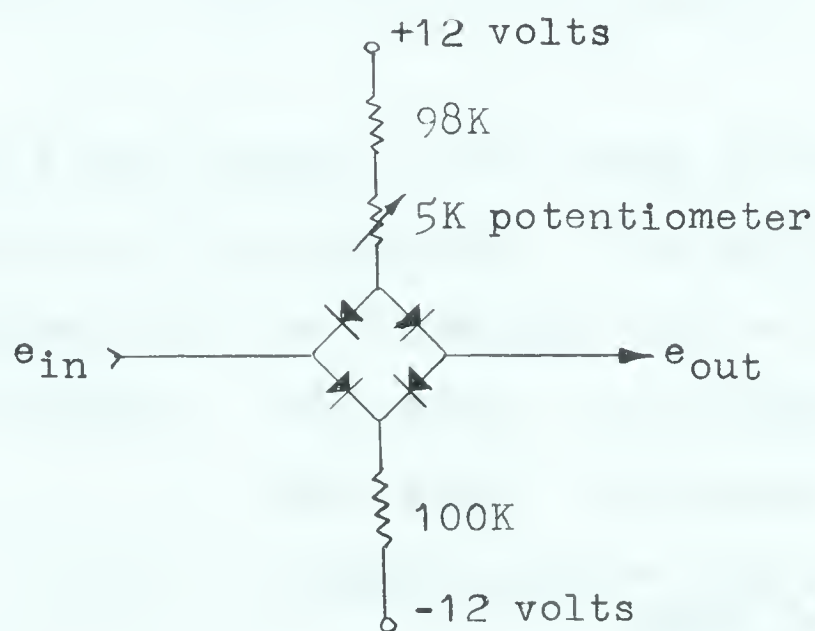


FIGURE B-3. MODIFIED DIODE BRIDGE SWITCH.

The bridge can be nulled almost exactly giving an integrated error for ten seconds of less than 0.03 volts. Better accuracy could be obtained if all diode bridges were placed in an oven thereby reducing drift due to changes in temperature.



## APPENDIX C. ELECTRONIC COMPARATOR

In order to achieve faster switching for the diode bridges, an electronic comparator was built up. A block diagram for the comparator is given in Figure C-1.

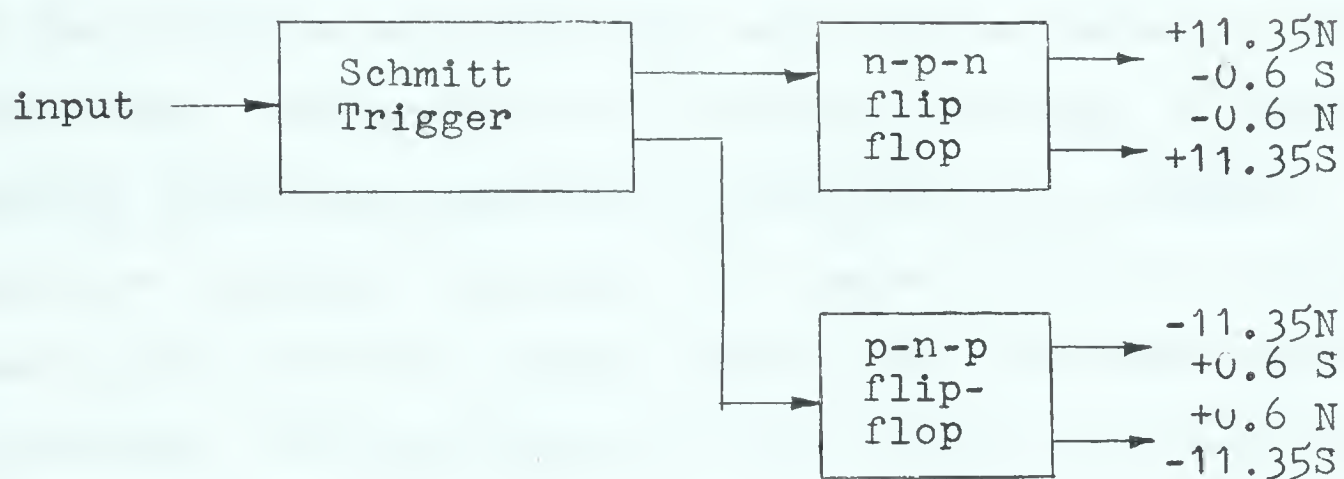


FIGURE C-1. COMPARATOR BLOCK DIAGRAM.

The symbols N and S refer to the normal and switched states respectively of the comparator. The performance of each of the components of the comparator are as follows:

Schmitt Trigger: Rise time -  $.2\mu$ seconds

Fall time -  $.2\mu$ seconds

Trigger point  $\approx 1.54$  volts (see Table C-1 for the trigger point of each comparator)

An emitter follower input stage with a clamping diode on the input allows the input voltage to the comparator to swing from -200 to +200 volts.

Direct coupling is used from the Schmitt trigger to the



bistable multivibrators. This is done to insure the correct state of each of the multivibrators. Clamping diodes are used on the collectors of the transistors to prevent the excursion into the reverse biased region from exceeding approximately  $\frac{1}{2}$  volt. This negative excursion into the reverse biased region is required to insure that the diode bridges driven by the bistable multivibrators are reverse biased or non-conducting. Otherwise if the collector voltage of the conducting transistor approached zero volts (or a slightly positive voltage) this would not insure that this voltage applied to the diode bridges would stop them from completely conducting. The performance of the bistable multivibrators is as follows:

Rise time =  $0.4\mu\text{seconds}$

Fall time =  $0.75\mu\text{seconds}$

A test to check the performance or the balance of the voltage outputs from the n-p-n and p-n-p bistable multivibrators is given in Figure C-2.

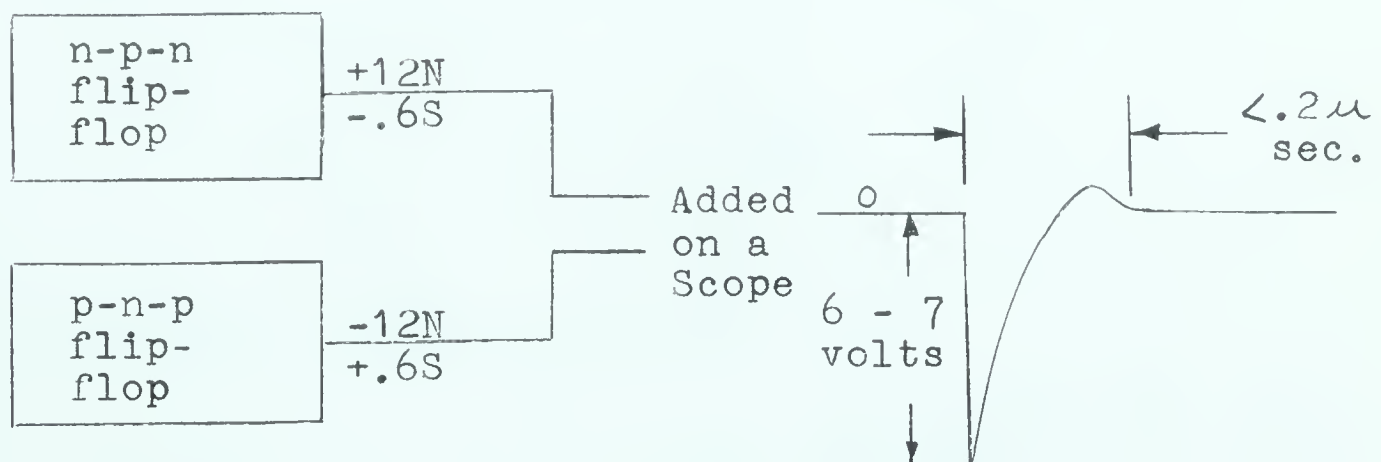


FIGURE C-2. PERFORMANCE TEST FOR COMPARATOR.



As shown in Figure C-2, the unbalance gives rise to a spike of six to seven volts with a duration of less than  $0.2\mu$  seconds. This switching effect causes an error less than 0.05 volts in a memory using a .001 ufd. capacitor. The switching level for each comparator is given in Table C-1. The wiring diagram for the comparator is given in Figure C-3.

Comparator	Switching Level	Comparator	Switching Level
1	0.85	7	1.25
2	1.58	8	1.45
3	1.60	9	1.08
4	1.75	10	1.25
5	1.40	11	1.30
6	1.25	12	1.24

TABLE C-1. SWITCHING LEVELS OF COMPARATORS



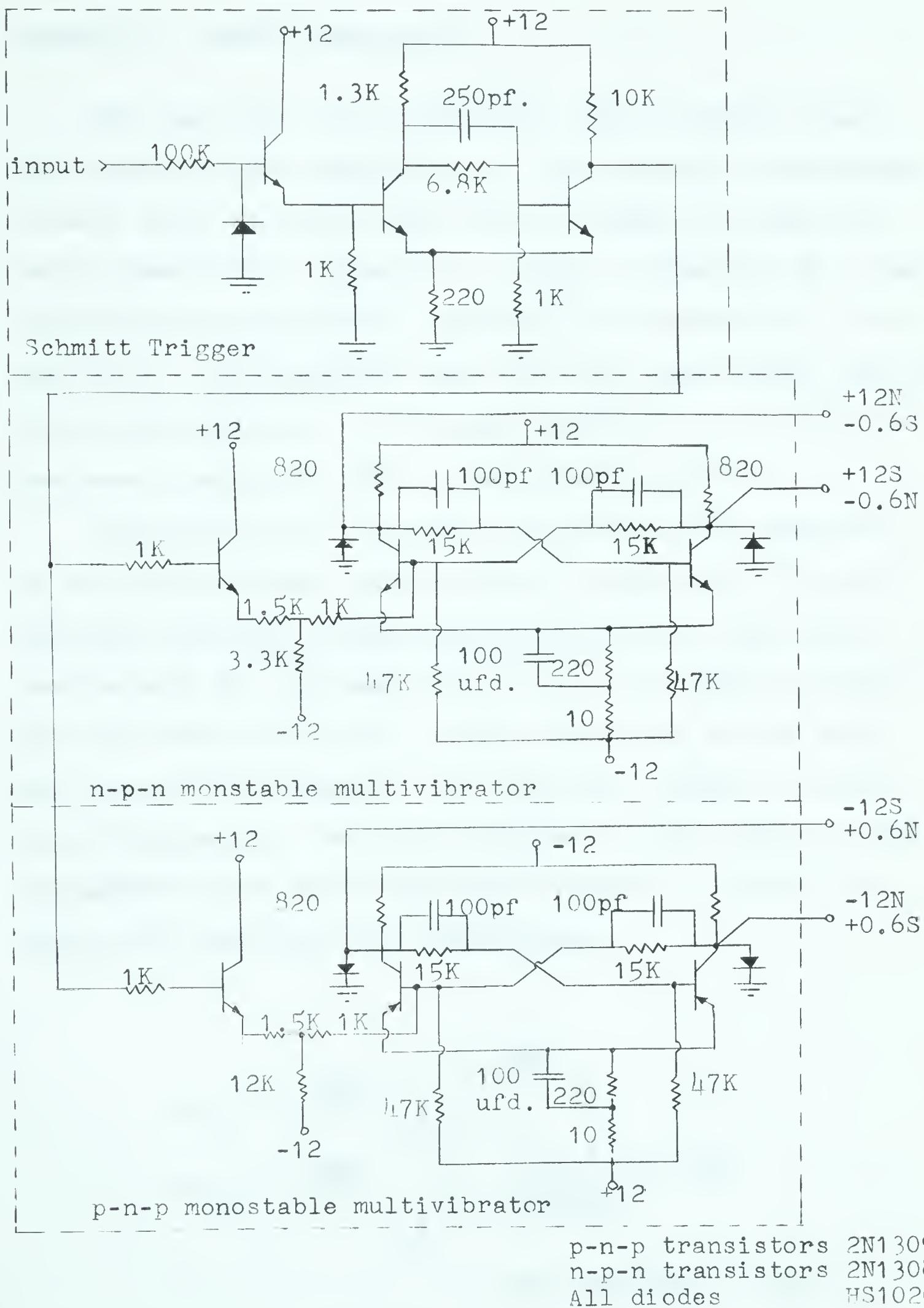


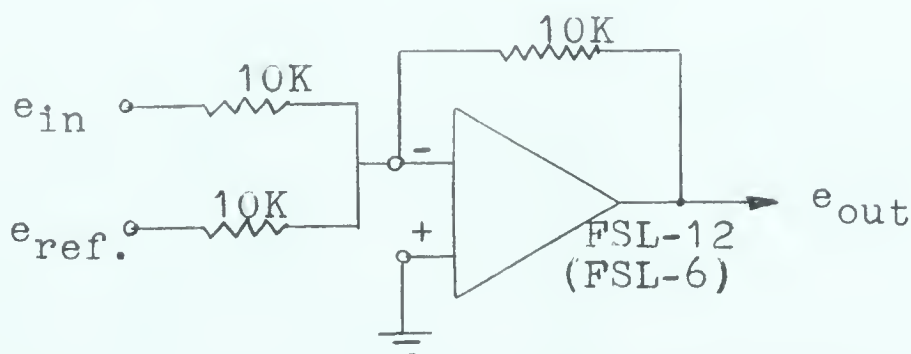
FIGURE C-3. WIRING DIAGRAM FOR COMPARATOR.



## APPENDIX D. NEXUS COMPARATOR

The need for a fast comparator with a variable dead zone arises in the application of synchronization techniques as well as in the real time read out scheme in iterative analog computation. The basis for this comparator is a fast transistorized operational amplifier, the Nexus FSL - 6 or the FSL-12. Provision was made for this operational amplifier to be used either as a unity-gain inverter or as a variable dead-zone comparator with limiters on the output.

When used as an inverter, the circuit will work well up to 100 kilocycles. When used as a comparator, the rise and fall times are in the vicinity of four to five microseconds with the limiters set to limit the output voltage to plus and minus five volts. With a dead zone of one volt, this circuit will operate in a reliable fashion to about forty kilocycles. The wiring diagram for the switch in the "Inverter" and in the "Comparator" position is given in Figure D-1 and Figure D-2 respectively.



All resistors 1 watt.

FIGURE D-1. NEXUS OPERATIONAL AMPLIFIER USED AS AN INVERTER.



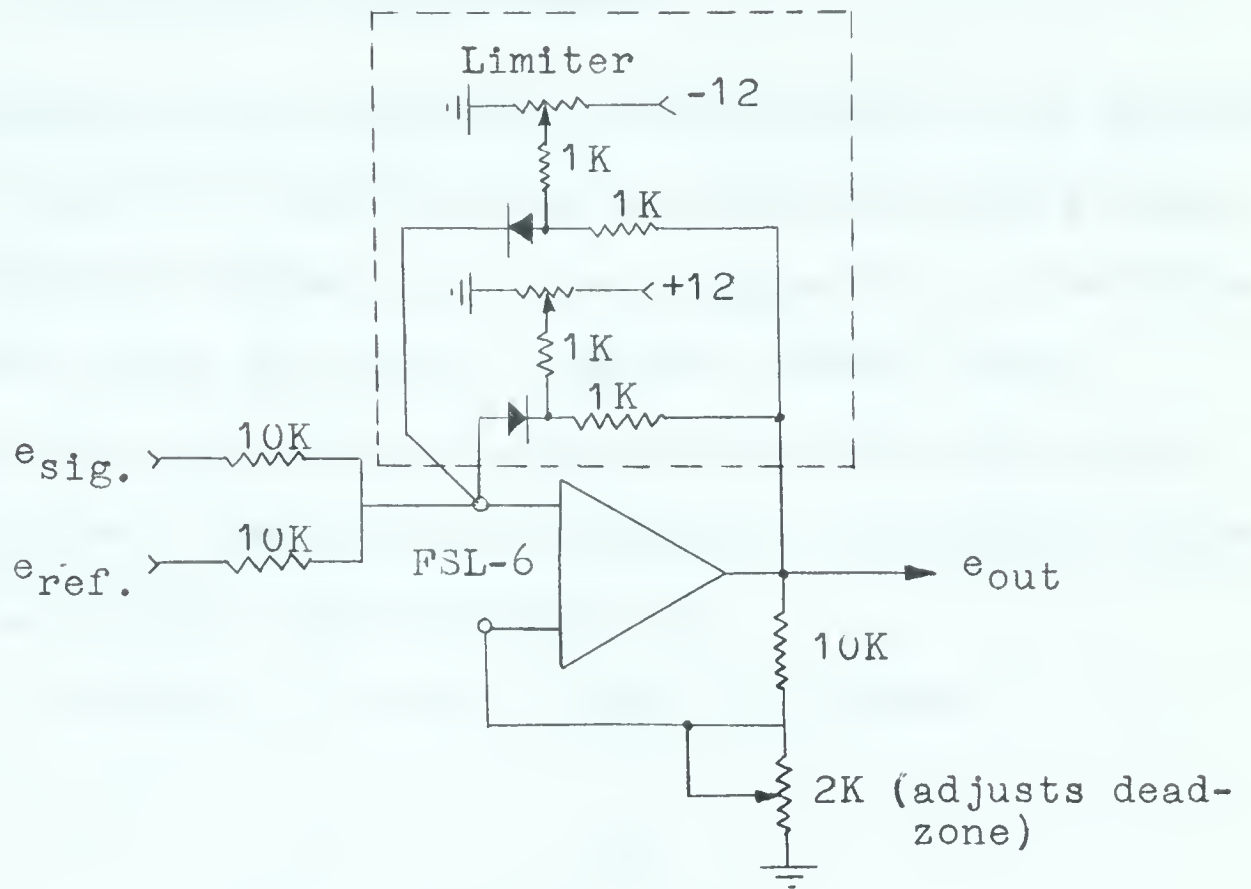


FIGURE D-2. WIRING DIAGRAM FOR VARIABLE DEAD-ZONE COMPARATOR.



## APPENDIX E. MONOSTABLE MULTIVIBRATOR

The purpose of the monostable multivibrator is to provide an inhibit pulse of fixed duration in iterative analog computation. Figure E-1 gives the wiring diagram for a monostable multivibrator which will give a negative inhibit pulse. Figure E-2 gives the wiring diagram for one that will give a positive pulse. The width or duration of the inhibit pulse in each case is given approximately by;

$$T = 82K \times C \times \ln 2 = 57.4C \times 10^3 \text{ seconds.}$$

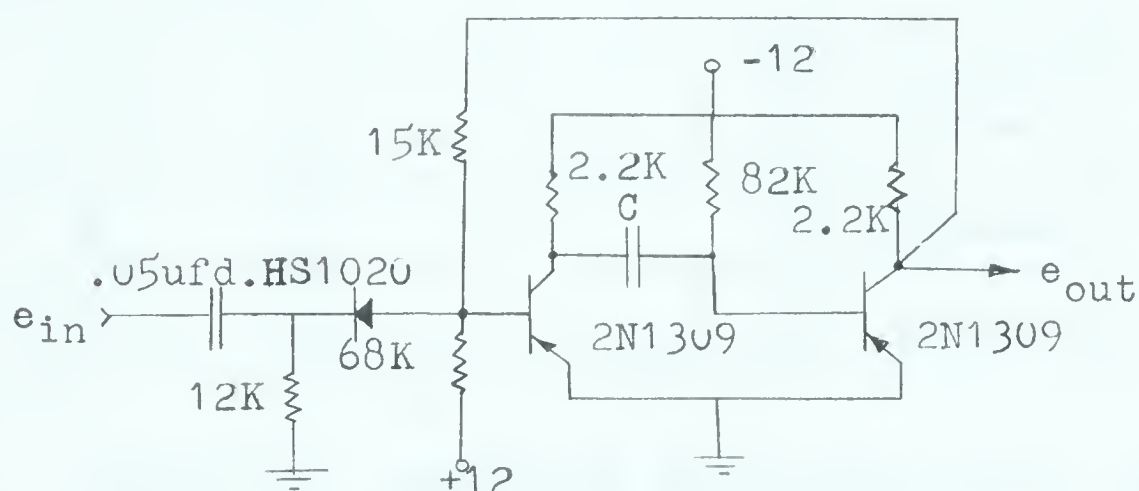


FIGURE E-1. MONOSTABLE MULTIVIBRATOR FOR NEGATIVE INHIBIT PULSE.

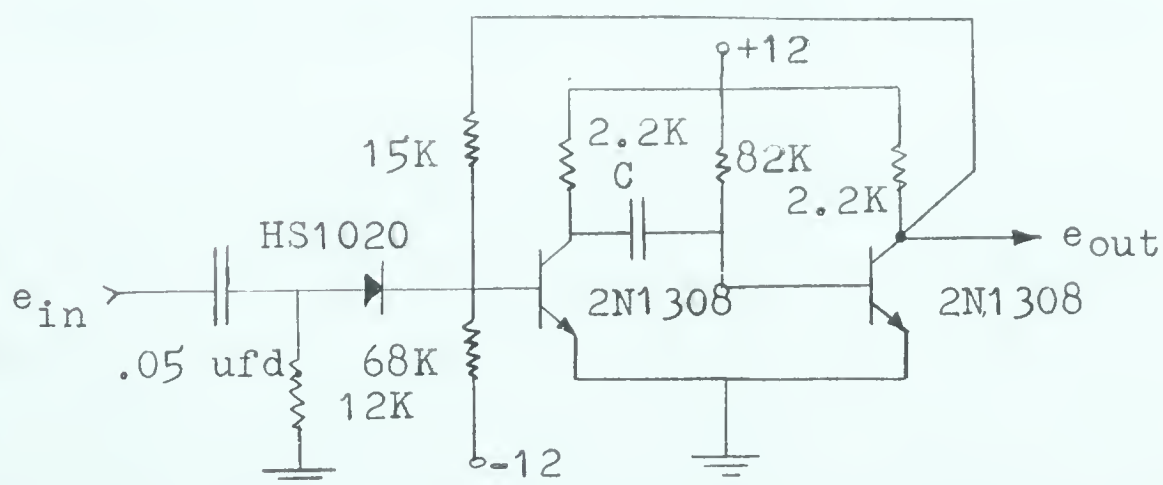


FIGURE E-2. MONOSTABLE MULTIVIBRATOR FOR POSITIVE INHIBIT PULSE.



Table E-1 gives a tabulation of the various monostable multivibrators used in the Iterative Control Unit. Duration of the inhibit pulse and the voltage output in the normal state for each is given.

Monostable Multivibrator	Sign of Inhibit Pulse	Pulse Duration	Voltage in Normal State
1	+	8.4 millisec.	+0.07
2	-	28 millisec.	-0.07
3	+	11 millisec.	+0.06
4	-	33 millisec.	-0.08
5	+	5 millisec.	+0.04
6	-	14 millisec.	-0.08

TABLE E-1. MONOSTABLE MULTIVIBRATORS IN THE  
ITERATIVE CONTROL UNIT.



## APPENDIX F. SWITCHING ARRANGEMENT FOR INTEGRATORS.

When it is required to switch the mode of an integrator, one possible scheme is the removal of the pot-set relay and then placing a diode switch in the summing junction. However, even a small unbalance in the diode bridge after being integrated will produce a significant error. An alternate scheme was devised to switch the integrator into the initial condition, operate, and hold modes which is not critically dependent on diode bridge balance. The scheme employed is shown in Figure F-1.

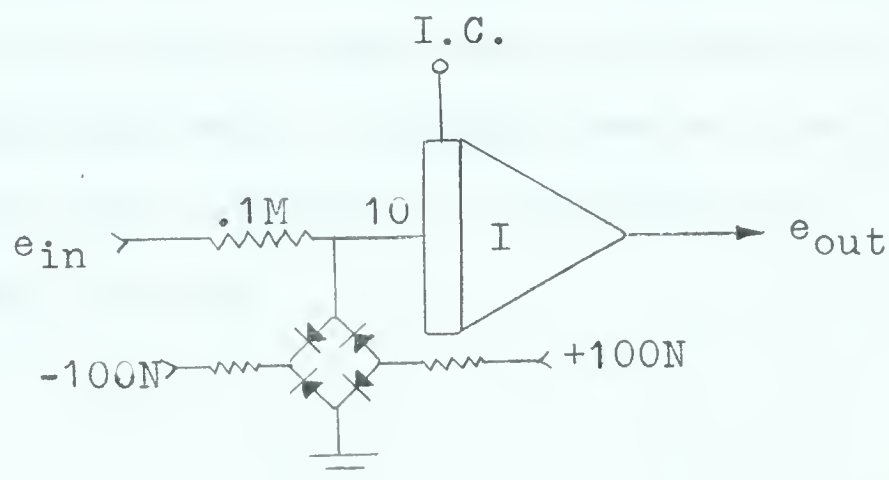


FIGURE F-1. SWITCHING ARRANGEMENT FOR INTEGRATORS.

With the indicated applied voltage, the diode bridge will conduct heavily bringing the input voltage to the integrator I very near zero volts. This will correspond to placing the integrator either in the initial condition or the hold mode. If the diode switch has the opposite polarity applied, it will be cut-off and I will have the property



of an integrator in the operate mode with a .2M input resistor. If the applied polarity to the bridge is again changed, causing it to conduct heavily, the integrator will now hold its last value.

The error of I while integrating is less than 0.04 volts in ten seconds. The drift while the integrator is placed in the hold mode is approximately .08 volts in ten seconds. The error however can be minimized even further if the "gain of 1" input on integrator I is used. With a gain of 1 used on integrator I, drift in the hold mode is less than 0.02 volts in 10 seconds. It is important to use as large a voltage as available to put the diode bridge into heavy conduction. This will help to limit the drift when the integrator is in the hold mode. Slightly better results in both the operate and hold modes can be obtained if a balanced diode bridge is used.



# APPENDIX G. NEWTON'S FORWARD DIFFERENCE INTERPOLATION FORMULA.

The Newton's Forward Difference Interpolation Formula for ten points is:

$$\begin{aligned}
 f(t) = & a + bt + \frac{ct(t-1)}{2!} + \frac{dt(t-1)(t-2)}{3!} + \frac{et(t-1)(t-2)(t-3)}{4!} + \\
 & \frac{gt(t-1)(t-2)(t-3)(t-4)}{5!} + \frac{ht(t-1)(t-2)(t-3)(t-4)(t-5)}{6!} \\
 & + \frac{it(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)}{7!} + \\
 & \frac{jt(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)(t-7)}{8!} + \\
 & \frac{kt(t-1)(t-2)(t-3)(t-4)(t-5)(t-6)(t-7)(t-8)}{9!} \dots (G-1)
 \end{aligned}$$

The values of the stored function at equal intervals of  $t$  will be designated as indicated in Figure G-1.

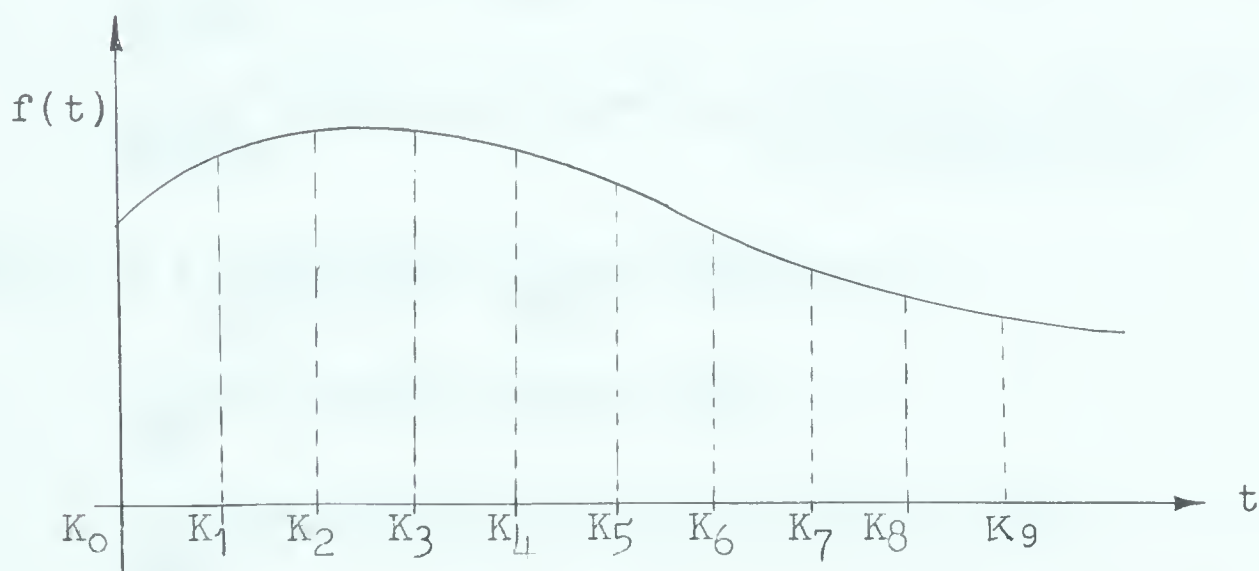


FIGURE G-1. NEWTON'S METHOD FOR 10 POINTS.

The derivatives of all orders for expression (G-1) will now be derived.



$$\begin{aligned}f'(t) = & b + \frac{c}{2}(2t-1) + \frac{d}{6}(3t^2-6t+2) + \frac{e}{24}(4t^3-18t^2+22t-6) + \\& \frac{g}{120}(5t^4-40t^3+105t^2-100t+24) + \\& \frac{h}{720}(6t^5-75t^4+340t^3-675t^2+540t-120) + \\& \frac{i}{5040}(7t^6-126t^5+875t^4-2940t^3+4875t^2-3528t+720) + \\& \frac{j}{40,320}(8t^7-196t^6+1932t^5-9800t^4+27,076t^3-39,396t^2+26,136t-5040) \\& + \frac{k}{362,880}(9t^8-288t^7+3822t^6-27,216t^5+112,245t^4-269,136t^3+354,372t^2 \\& \quad -219,168t+40,320)\end{aligned}$$

$$\begin{aligned}f''(t) = & c + \frac{d}{6}(6t-6) + \frac{e}{24}(12t^2-36t+22) + \frac{g}{120}(20t^3-120t^2+210t-100) + \\& \frac{h}{720}(30t^4-300t^3+1020t^2-1350t+548) + \\& \frac{i}{5040}(42t^5-630t^4+3500t^3-8820t^2+9750t-3528) + \\& \frac{j}{40,320}(56t^6-1176t^5+9660t^4-39,200t^3+81,228t^2-78,792t+26,136) + \\& \frac{k}{362,880}(72t^7-2016t^6+22,932t^5-136,080t^4+448,980t^3-807,408t^2+ \\& \quad 708,744t-219,168)\end{aligned}$$

$$\begin{aligned}f'''(t) = & d + \frac{e}{24}(24t-36) + \frac{g}{120}(60t^2-240t+210) + \\& \frac{h}{720}(120t^3-900t^2+2,040t-1350) + \\& \frac{i}{5040}(210t^4-2520t^3+10,500t^2-17,640t+9,750) + \\& \frac{j}{40,320}(336t^5-5880t^4+38,640t^3-117,600t^2+162,456t-78,792) + \\& \frac{k}{362,880}(504t^6-12,096t^5+114,660t^4-544,320t^3+346,940t^2- \\& \quad 1,614,816t+709,744)\end{aligned}$$



$$f^{iv}(t) = e + \frac{g(120t-240)}{120} + \frac{h(360t^2-1800t+2040)}{720} +$$

$$\frac{i(840t^3-7560t^2+21,000t-17,640)}{5040} +$$

$$\frac{j(1680t^4-23,520t^3+115,920t^2-235,200t+162,456)}{40,320} +$$

$$\frac{k(3,024t^5-60,480t^4+458,640t^3-1,632,960t^2+2,693,880t-1,614,816)}{362,880}$$

$$f^v(t) = g + \frac{h(720t-1800)}{720} + \frac{i(2520t^2-15,120t+21,000)}{5040} +$$

$$\frac{j(6720t^3-70,560t^2+231,840t-235,200)}{40,320} +$$

$$k(15,120t^4-241,920t^3+1,375,920t^2-2,265,920t+2,693,880)$$

$$f^{vi}(t) = h + \frac{i(5040t-15,120)}{5040} + \frac{j(20,160t^2-141,120t+231,840)}{40,320} +$$

$$\frac{k(60,480t^3-725,760t^2+2,751,840t-2,265,920)}{362,880}$$

$$f^{vii}(t) = i + \frac{j(40,320t-141,120)}{40,320} + \frac{k(181,440t^2-1,451,520t+2,751,840)}{362,880}$$

$$f^{viii}(t) = j + \frac{k(362,880t-1,451,520)}{362,880}$$

$$f^{ix}(t) = k$$

By forming a table as given in reference 1, we may find the coefficients a,b,c,...k. Table G-1 is drawn up for values a to e in order to demonstrate the manner in which these coefficients may be found.



$\Delta_1=a$	$\Delta_2=b$	$\Delta_3=c$	$\Delta_4=d$	$\Delta_5=e$
$K_0$				
$K_1$	$K_1-K_0$			
$K_2$	$K_2-K_1$	$K_2-2K_1+K_0$	$K_3-3K_2+3K_1-K_0$	
$K_3$	$K_3-K_2$	$K_3-2K_2+K_1$	$K_4-3K_3+3K_2-K_1$	$K_4-4K_3+6K_2-4K_1+K_0$
$K_4$	$K_4-K_3$	$K_4-2K_3+K_2$		

TABLE G-1. DETERMINATION OF COEFFICIENTS a to e.

Coefficients a to i are given in Table G-2 in terms of the value of the function at the sampled points  $K_i$ .

$$a = K_0$$

$$b = K_1 - K_0$$

$$c = K_2 - 2K_1 + K_0$$

$$d = K_3 - 3K_2 + 3K_1 - K_0$$

$$e = K_4 - 4K_3 + 6K_2 - 4K_1 + K_0$$

$$g = K_5 - 5K_4 + 10K_3 - 10K_2 + 5K_1 - K_0$$

$$h = K_6 - 6K_5 + 15K_4 - 20K_3 + 15K_2 - 6K_1 + K_0$$

$$i = K_7 - 7K_6 + 21K_5 - 35K_4 + 35K_3 - 21K_2 + 7K_1 - K_0$$

TABLE G-2. COEFFICIENTS a TO i.

In order to employ Newton's Iterative Method the value of the derivatives of all orders of the approximation function must be known at  $t = 0$ . The results for five points are given in Table G-3.



$$f(t) = a + bt + \frac{ct(t-1)}{2!} + \frac{d(t^3-3t^2+2t)}{3!} + \frac{e(t^4-6t^3+11t^2-6t)}{4!}$$

$$f(0) = a$$

$$f'(t) = b + \frac{c(2t-1)}{2!} + \frac{d(3t^2-6t+2)}{3!} + \frac{e(4t^3-18t^2+22t-6)}{4!}$$

$$f'(0) = b - .5c + .333d - .25e$$

$$f''(t) = c + \frac{d(6t-6)}{3!} + \frac{e(12t^2-36t+22)}{4!}$$

$$f''(0) = c - d + .916e$$

$$f'''(t) = d + \frac{e(24t-36)}{4!}$$

$$f'''(0) = d - 1.5e$$

$$f^{IV}(t) = e$$

$$f^{IV}(0) = e$$

TABLE G-3. VALUES OF  $f^i(0)$ .





**B29828**